1ØØ DICAS PARA MSX

TÉCNICAS E TRUQUES DE PROGRAMAÇÃO

RENATO DA SILVA OLIVEIRA

1ØØ DICAS PARA MSX

TÉCNICAS E TRUQUES DE PROGRAMAÇÃO

AUTORES:

Henrique de Figueredo Luz
Luis Tarcísio de Carvalho Jr.
Milton Maldonado Jr. (The Pilot)
Pierluigi Piazzi
Renato da Silva Oliveira
Rubens Pereira Silva Jr.

3ª EDIÇÃO
1989

EXPEDIENTE:

Coordenação Editorial: PIERLUIGI PIAZZI
Coordenação Didática : BETTY FROMER PIAZZI
Produção Editorial : ROSA KOGAN FROMER
Editoração : RENATO DA SILVA OLIVEIRA
Ilustrações : DURVALY ODILON NICOLETTI

ALEPH Publicações e
Assessoria Pedagógica Ltda
R. Dr. Luiz Migliano 1110 cj.301
05711 São Paulo SP
Caixa Postal 20707 CEP 01498
Tel: (011) 843-3202

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Oliveira, Renato da Silva, 1960-
O51c 100 dicas para MSX : técnicas e truques de programação / Renato da Silva Oliveira. -- 1. ed. -- São Paulo : Aleph, 1988.
(Coleção MSX)

1. MSX (Computadores) - Programação I. Título. II. Série.

88-0118 CDD-001.642

Índices para catálago sistemático:

1. MSX : Computadores : Programação : Processamento de dados 001.642

CEM DICAS PARA O MSX

SUMARIO

APRESENTACAO 6

1 - DICAS SOBRE O TECLADO 7

2 - DICAS PARA USAR O VIDEO 21

3 - DICAS SONORAS 88

4 - DICAS PARA CASSETE 105

5 - DICAS PARA IMPRESSORA 117

6 - DICAS PARA O DRIVE 134

7 - DICAS DE PROCESSAMENTO 153

8 - NOTAS SOBRE A BIBLIOGRAFIA RECOMENDADA .. 191

APRESENTAÇÃO

A grande maioria dos usuários de microcomputadores MSX nunca teve outro micro antes. Apesar de terem entrado no mundo da computação pela porta da frente, é provável que esses usuários tenham alguma dificuldade inicial para programar suas máquinas. Poucos deles têm, de imediato, uma visão muito clara dos poderosos recursos de que dispõem.

Este livro contém mais de cem dicas de programação já prontas para serem usadas. Elas permitem um aproveitamento muito maior dos recursos dos MSX mesmo pelos usuários iniciantes e, certamente, acrescentam conhecimentos valiosos aos programadores mais experientes.

Apesar de serem geralmente independentes umas das outras, as dicas estão agrupadas em 7 capítulos, de acordo com a função a que elas se destinam.

O leitor poderá usá-las em outros programas para otimizá-los ou obter resultados específicos.

Cada dica está apresentada de forma bem prática e resumida de modo a tornar seu uso imediato muito fácil. As explicações nem sempre são detalhadas mas, sempre que possível, faz-se referência a textos de outros livros onde o assunto é comentado mais extensamente.

Apesar de muitas dicas serem programas em Linguagem de Máquina, optamos pelo uso exclusivo da linguagem BASIC para gerá-las. Isso torna a digitação mais fácil e o uso mais imediato. Todos os programas estão em BASIC, mas mesmo assim, para diminuir a ocorrência de erros de digitação, junto a cada listagem pode-se encontrar em "vídeo inverso" sua SOMA SINTÁTICA. As explicações sobre a produção e uso dessa "soma" são dadas na dica 7.I (página 178) e ela é, portanto, a primeira dica que deve ser lida.

Lembre-se que a SOMA TOTAL de um programa, mesmo coincidindo com a apresentada no livro, não elimina totalmente a ocorrência de erros de digitação, mas apenas a reduz. Deve-se também considerar que a soma que apresentamos foi obtida com um EXPERT 1.1 e há casos em que outros tipos de MSX produzem somas diferentes.

Esperamos que este livro possa abrir novos horizontes aos programadores MSX; tanto aos iniciantes quanto aos que não têm tempo suficiente para descobrir sozinhos os incontáveis "macêtes" dessas máquinas maravilhosas.

DICAS SOBRE O TECLADO

As dicas deste capítulo abordam preponderantemente assuntos relativos a manipulação do teclado nos micros MSX.

Uma vez que o principal meio de entrada de dados para a UCP do micro é o teclado, a utilidade destas dicas é evidente.

1.1 - Carregando o Buffer do Teclado 8
1.2 - Limpando o Buffer do Teclado 9
1.3 - Programando as teclas de funções 10
1.4 - Restabelecendo as teclas de funções .. 11
1.5 - Checando as teclas especiais 12
1.6 - Travando a tecla CAPS LOCK 13
1.7 - Usando o click do teclado 14
1.8 - Reprogramando todo o teclado 15
1.9 - Usando a barra de espaços 18
1.A - Usando as teclas de setas 19
1.B - Uso da instrução ON KEY GOSUB 20

1.1 - CARREGANDO O BUFFER DO TECLADO

O MSX reserva uma área (buffer) de 40 bytes na memória RAM para armazenar temporariamente os dados digitados através do teclado. Se esses dados forem comandos, serão executados na sequência em que estiverem no buffer. Pode-se carregar comandos no buffer do teclado através de programas em BASIC ou em ASSEMBLY a fim de gerar efeitos especiais. Por exemplo, para fazer com que um programa em BASIC seja carregado de fita cassete e seja automaticamente executado, basta usar o programa apresentado a seguir.

```
100 SCREEN 1:X$=CHR$(34):Y$=CHR$(13)          8CC
110 A$="CLOAD"+X$+"CAS:"+X$+":RUN"+Y$        159E
120 EN=&HFBF0                                1917
130 FOR F=1 TO LEN(A$)                       20BE
140  CH=ASC(MID$(A$,F,1))                    290F
150  POKE EN,CH                              2D18
160  EN=EN+1                                 311F
170 NEXT F                                   34D7
180 X=65536!+&HFBF0+LEN(A$)                  40B8
190 Y=X-256*INT(X/256)                       4ADF
200 POKE &HF3F8,Y                            4F80
210 POKE &HF3F9,INT(X/256)                   5890
220 POKE &HF3FA,&HF0                         5EA3
230 POKE &HF3FB,&HFB                         64D7

                        TOTAL   =            64D7
```

Os dados a serem inseridos no buffer estão na variável A$. A variável EN armazena o endereço do buffer a ser preenchido. A variável CH armazena o caractere de A$ a ser inserido no endereço EN do buffer. O endereço &HFBF0 é o início do buffer do teclado (KEYBUFF).

Os endereços &HF3F8 e &HF3F9 armazenam o próximo endereço a ser preenchido no buffer do teclado (PUTPNT) e são preenchidos de modo a apontarem para o endereço subsequente ao do fim da mensagem inserida.

Os endereços &HF3FA e &HF3FB armazenam o último endereço do buffer lido pelo micro (GETPNT) e são posicionados de modo a apontarem para o endereço do primeiro caractere inserido (&HFBF0).

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - páginas 40, 41, 42 e 143.

1.2 - LIMPANDO O BUFFER DO TECLADO

Observe o programa a seguir:

```
10 SCREEN 0,,0:LOCATE,,0:KEY OFF                    502
20 PRINT " Entre uma escolha:":PRINT                C83
30 PRINT " [ 1 ] Opção 1 ...."                     141F
40 PRINT " [ 2 ] Opção 2 ...."                     1EAA
50 PRINT " [ 3 ] Opção 3 ...."                     2778
60 PRINT:PRINT:PRINT                               2884
70 B$=INKEY$:IF LEN(B$)<1 THEN 70                  3802
80 FOR F=1 TO 500 : NEXT F : BEEP                  4407
90 PRINT "OPÇÃO ";B$;" ESCOLHIDA !"                5800
100 A$=INPUT$(1)                                   5F13
110 PRINT A$                                       6378

                                   TOTAL    =      6378
```

Note que ao fazer a primeira opção, se a tecla da escolha for pressionada por muito tempo ou se mais de uma tecla for pressionada, a segunda opção também será feita. Isso ocorre porque o buffer do teclado fica carregado com os caracteres digitados até que eles sejam usados.

No BIOS do MSX existe uma rotina que pode ser útil nessas situações: a KILLBUFF. Sempre que é executada, ela limpa o buffer do teclado. Para chamá-la, basta usar as instruções:

```
DEFUSR0 = &H0156 : POKE 0,USR0(0)
```

Experimente inserir a linha a seguir no programa anterior e depois execute-o novamente.

```
95 DEFUSR0 = &H0156 : POKE 0,USR0(0)
```

Você notará que a segunda opção não mais será atrapalhada por digitações acidentais.

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX vol. 2 - página 110.
Aprofundando-se no MSX - página 159.

1.3 - PROGRAMANDO AS TECLAS DE FUNÇÕES

As teclas de funções podem ser facilmente redefinidas para atender às necessidades específicas de um programa.

A forma mais direta de reprogramá-las é usando a instrução KEY. Entretanto pode-se também redefinir as teclas de funções diretamente na memória RAM.

Experimente executar o seguinte programa e depois pressione a tecla F1.

```
10 FOR F=&HF87F TO &HF87F+38
20   POKE F,ASC("A")
30 NEXT F
```

Com isso a tecla F1 passou a ter uma sequência de 39 caracteres "A". Lembre-se que com a instrução KEY podemos inserir no máximo 15 caracteres em cada tecla de função. Com este recurso, podemos atribuir a uma única tecla até 39 caracteres, entretanto deve-se tomar alguns cuidados, pois o conteúdo das demais teclas podem ser alterados. Experimente digitar a tecla F2 após ter executado o programa acima. Você verá que seu conteúdo foi alterado pelo programa.

O que acontece é que existem 160 bytes da RAM, reservados a partir do endereço &HF87F (FNKSTR), para armazenar os textos das teclas de funções. Cada tecla tem seu texto começando sempre num mesmo endereço e o número máximo de caracteres atribuíveis a uma única tecla de função é 39. O 40º caractere da área de texto de uma tecla é sempre um 0.

Você pode também atribuir às teclas de funções sequências de caracteres de controle. Por exemplo, digite a instrução a seguir e depois pressione a tecla F1.

```
KEY 1,CHR$(7)+CHR$(28)+CHR$(8)+CHR$(7)+C
HR$(127)+CHR$(9)+CHR$(11)
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - página 81.
Curso de Basic v.1 - páginas 21 e 22.
Aprofundando-se no MSX - página 50.
Programação Avançada em MSX - página 145

1.4 - RESTABELECENDO AS TECLAS DE FUNÇÕES

As teclas de funções podem ser facilmente programadas. Entretanto, após usar um programa que as redefina, pode ser necessário reinicializá-las com as funções originais. Para isso pode-se simplesmente programar tecla por tecla novamente ou chamar uma rotina do BIOS (INITFNK em &H003E) que se encarrega de fazer isso automaticamente. Para executar essa rotina do BIOS basta digitar a seguinte instrução:

```
DEFUSR0 = &H3E : POKE 0,USR0(0)
```

Observe o programa exemplo a seguir. Ele redefine as teclas de funções e logo a seguir restabelece seus conteúdos originais.

```
10 SCREEN 0 : KEY ON                          328
20 FOR F=1 TO 10                              602
30   KEY F,"NOVA !"                           A87
40 NEXT F                                     8FE
50 SCREEN 0                                   1C5
60 PRINT,," TECLAS REDEFINIDAS !"             1890
70 PRINT,," PRESSIONE RETURN !"               190F
80 A$=INPUT$(1)                               1000
90 DEFUSR0 = &H3E : X = USR0(0)               29F9
100 SCREEN 0                                  20FF
110 PRINT,," TECLAS RESTABELECIDAS !"         4646

                              TOTAL =         4646
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - página 156.

1.5 - CHECANDO AS TECLAS ESPECIAIS

Nos micros MSX existem várias maneiras de verificarmos se alguma tecla especial está ou não pressionada. Uma delas é consultando as variáveis do sistema OLDKEY (&HFBDA) ou NEWKEY (&HFBE5). O programa listado a seguir ilustra esse procedimento, checando especificamente se a tecla LGRA (ou GRAPH) está pressionada.

Ao ser executado o programa lê o conteúdo da região da memória RAM que vai do endereço &HFBDA até o endereço &HFBEF, mostrando na tela seus conteúdos em binário. Experimente pressionar algumas teclas (menos ^STOP!). Você verá que a cada tecla pressionada corresponde um bit de algum dos bytes dessa região. Para checar qualquer tecla, portanto, basta verificar se o bit está ou não em "0". Experimente pressionar a tecla LGRA (ou GRAPH). A configuração que corresponde à ela é o valor 251 (ou 11111011, em binário) no byte &HFBE0. Observe como a linha 250 faz o teste para ver se ela está ou não pressionada e tente alterar o programa para que ele teste se a tecla RGRA (ou CODE) está pressionada.

```
100 REM
110 REM LE OLDKEY  E NEWKEY
120 REM
130 SCREEN 0
140 LOCATE 0,5,0
150 FOR F=&HFBDA TO &HFBE4
160   PRINT "&H";HEX$(F);"=> ";
170   PRINT RIGHT$("00000000"+BIN$(PEEK(
F)),8);
180   PRINT "   &H";HEX$(F+11);"=> ";
190   PRINT RIGHT$("00000000"+BIN$(PEEK(
F+11)),8)
200 NEXT F
210 PRINT:PRINT:PRINT
220 REM
230 REM TESTA A TECLA LGRA
240 REM
250 IF PEEK(&HFBE0)=(PEEK(&HFBE0)AND251)
THEN PRINT "LGRA PRESSIONADA"ELSEPRINT"
"
260 GOTO 140
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 83 a 86.

1.6 - TRAVANDO A TECLA CAPS LOCK

Existe uma variável do sistema (CAPST, em &HFCAB) que indica o estado da tecla CAPS LOCK. Se CAPS LOCK está ativa, essa variável contém 255 ou algum outro valor maior que zero, e se inativa, contém 0. Cada vez que a tecla CAPS LOCK é pressionada os bits da variável CAPST são complementados de modo que seu valor é sempre 0 ou 255.

Para simular o pressionamento de CAPS LOCK basta "pokear" 255 ou 0 em CAPST.

Para travar a CAPS LOCK, deixando-a ativa, basta pokear qualquer valor maior que 0 e menor que 255 em CAPST, pois assim, mesmo que CAPS LOCK seja pressionada, a complementação dos bits de CAPST produzirá um valor também maior que 0 e menor que 255.

Experimente digitar e usar o programa a seguir. Após executá-lo, digite algumas letras usando as teclas CAPS LOCK e SHIFT. Você notará que elas ficaram praticamente inoperantes para as letras.

```
10 SCREEN 0 : WIDTH 38
20 POKE &HFCAB,1
30 PRINT,,"   DIGITE ALGUMAS LETRAS"
40 PRINT"   (com ou sem SHIFT):"
50 PRINT : PRINT
60 A$=INPUT$(1)
70 PRINT A$;
80 GOTO 60
```

Para fazer com que a tecla CAPS LOCK volte a funcionar normalmente, comande:

```
POKE &HFCAB,0
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - página 50.
Programação Avançada em MSX - página 146.
Coleção de Programas para MSX vol 2 - páginas 51 e 55.

1.7 - USANDO O CLICK DO TECLADO

O click do teclado pode ser usado para muitas e diferentes aplicações, deixando livre para outros usos o PSG.

Para gerar o click do teclado por software, é necessário acessar o hardware da máquina com comandos OUT. Veja o programa a seguir.

```
10 SCREEN 1                       10E
20 KEY OFF                        2F5
30 COLOR 1,4                      451
40 LOCATE 32*RND(1),24            941
50 PRINT "*"                      A44
60 OUT &HAA,&HFF                  1056
70 OUT &HAA,&H7F                  1091
80 GOTO 40                        1368

                  TOTAL   =       1368
```

Cada vez que as linhas 60 e 70 são executadas um "click" é gerado. Você pode usar isto para sonorizar seus programas mesmo sem usar o PSG.

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX vol 2 - página 15.
Aprofundando-se no MSX - página 87.

1.8 - REPROGRAMANDO TODO O TECLADO

Agora seu MSX também pode ter um "teclado inteligente" como nos velhos tempos do Apple e do Sinclair. O programa a seguir instala em seu teclado as principais palavras reservadas do Basic sem perder sua função original.

A redefinição atinge as letras maiúsculas e minúsculas, totalizando 52 palavras reservadas. No que é possível, as teclas seguem o padrão Sinclair (A=NEW, K=LIST, etc). Palavras do Sinclair que não existem no MSX (PAUSE, SCROLL, etc) são trocadas por outras do último.

Para usar o programa, digite-o e comande RUN. Após a execução deverão aparecer os códigos das teclas e as palavras correspondentes. Experimente então teclar SELECT e em seguida a tecla A. Você deverá obter "NEW". Tecle agora SELECT+"a" para obter "FRE(". Tente então outras combinações. O que acontece quando a tecla após o SELECT não é uma letra?

```
1000 REM ------------------------           646
1010 REM TOKEN V. 1.0 BY THE PILOT          F7E
1020 REM      JANEIRO DE 1988               1509
1030 REM ------------------------           205C
1040 REM                                    222E
1050 DATA 21,9B,D1,CD,CE,D1,21,16           30E9
1060 DATA D0,22,A5,FD,3E,C3,32,A4           480A
1070 DATA FD,AF,32,D7,D1,C9,4F,3A           5522
1080 DATA D7,D1,A7,79,20,05,FE,18           6B02
1090 DATA 28,49,C9,FE,41,38,3D,FE           8056
1100 DATA 5B,30,08,D6,40,47,21,72           88E0
1110 DATA D0,18,0E,FE,61,38,2D,FE           8E70
1120 DATA 7B,30,29,D6,60,47,21,08           9602
1130 DATA D1,7E,A7,23,20,FB,10,F9           9EEE
1140 DATA 3E,C9,32,A4,FD,7E,A7,28           A74F
1150 DATA 06,CD,A2,00,23,18,F6,3E           B9F9
1160 DATA C3,32,A4,FD,AF,32,D7,D1           C832
1170 DATA C1,C3,DA,08,4F,AF,32,D7           DEFC
1180 DATA D1,79,C9,3E,FF,32,D7,D1           F3CF
1190 DATA AF,C9,00,4E,45,57,00,42           396
1200 DATA 45,45,50,00,43,4F,4E,54           94B
1210 DATA 00,44,49,4D,20,00,52,45           C60
1220 DATA 4D,20,00,46,4F,52,20,00           17FE
1230 DATA 47,4F,54,4F,20,00,47,4F           2173
1240 DATA 53,55,42,20,00,49,4E,50           3003
1250 DATA 55,54,20,00,4C,4F,41,44           41A6
1260 DATA 20,00,4C,49,53,54,20,00           5276
```

```
1270 DATA 4C,4C,49,53,54,20,00,4D
1280 DATA 4F,54,4F,52,20,00,4E,45
1290 DATA 58,54,20,00,50,4F,4B,45
1300 DATA 20,00,50,52,49,4E,54,00
1310 DATA 50,53,45,54,20,28,00,52
1320 DATA 55,4E,00,53,41,56,45,20
1330 DATA 00,54,52,4F,4E,00,49,46
1340 DATA 20,00,43,4C,53,00,50,52
1350 DATA 45,53,45,54,20,28,00,43
1360 DATA 4C,45,41,52,00,52,45,54
1370 DATA 55,52,4E,00,45,4E,44,00
1380 DATA 00,46,52,45,28,00,49,4E
1390 DATA 4B,45,59,24,00,44,53,4B
1400 DATA 46,28,00,41,54,4E,28,00
1410 DATA 54,41,4E,28,00,53,47,4E
1420 DATA 28,00,41,42,53,28,00,53
1430 DATA 51,52,28,00,41,53,43,28
1440 DATA 00,56,41,4C,28,00,4C,45
1450 DATA 4E,28,00,55,53,52,00,33
1460 DATA 2E,31,34,31,35,39,32,37
1470 DATA 21,00,4E,4F,54,00,50,45
1480 DATA 45,4B,28,00,54,41,42,28
1490 DATA 00,53,49,4E,28,00,49,4E
1500 DATA 54,28,00,53,54,52,49,4E
1510 DATA 47,24,28,00,52,4E,44,28
1520 DATA 00,43,48,52,24,28,00,56
1530 DATA 41,52,50,54,52,28,00,43
1540 DATA 4F,53,28,00,45,58,50,28
1550 DATA 00,53,54,52,24,28,00,4C
1560 DATA 4E,28,00,0C,50,72,6F,67
1570 DATA 72,61,6D,61,20,65,73,63
1580 DATA 72,69,74,6F,20,70,6F,72
1590 DATA 3A,0D,0A,54,48,45,20,50
1600 DATA 49,4C,4F,54,20,65,6D,20
1610 DATA 4A,61,6E,65,69,72,6F,2F
1620 DATA 31,39,38,38,2E,00,7E,A7
1630 DATA C8,CD,A2,00,23,18,F7,00
1640 DATA FIM
1650 CLS:PRINT "CARREGANDO TOKEN"
1660 FOR I=&HD000 TO &HD1D8:READ A$:POKE
 I,VAL("&H"+A$):NEXT I
1670 DEFUSR=&HD000:A=USR(0):PRINT:PRINT
1680 FOR I=65 TO 90:PRINT "<SELECT>+"CHR
$(I);" = ";CHR$(24);CHR$(I):FOR T=0 TO 1
00:NEXT T:NEXT I
1690 FOR I=97 TO 122:PRINT "<SELECT>+"CH
R$(I);" = ";CHR$(24);CHR$(I):FOR T=0 TO
100:NEXT T:NEXT I
1700 END
```

É recomendável salvar o código binário para facilitar o uso do programa. Para isto digite:

```
BSAVE "TOKEN.BIN",&HD000,&HD1D8
```

ou

```
BSAVE "CAS:TOKEN",&HD000,&HD1D8
```

Para executar o programa, use o comando:

```
BLOAD "TOKEN.BIN",R
```

ou

```
BLOAD "CAS:",R
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 66 a 71 e capítulo 3.
Programação Avançado em MSX - capítulo 1 e apêndice 3.

1.9 - USANDO A BARRA DE ESPAÇOS

A barra de espaços no MSX pode ser usada de muitas e diferentes maneiras. No programa abaixo apresentamos um exemplo em que a barra é usada para controlar o fluxo do processamento usando a função STRIG.

```
10 SCREEN 0 : PRINT : PRINT SPC(6);          4B7
20 PRINT "DIGITE A BARRA DE ESPAÇO !"        E55
30 BEEP                                      FAB
40 IF NOT STRIG(0) THEN 30                   1616
50 PLAY "ABCDEFG"                            1914

                          TOTAL  =           1914
```

Uma outra forma de se controlar o fluxo do processamento é através das interrupções. Essa maneira é ilustrada pelo programa a seguir.

```
10 SCREEN 0 : PRINT : PRINT SPC(6);          4B7
20 PRINT "DIGITE A BARRA DE ESPAÇO !"        E55
30 STRIG(0) ON : ON STRIG GOSUB 60           1638
40 PLAY"V15ABCDEFGO7L32"                     1B0F
50 GOTO 50                                   1EE4
60 PLAY"C#" : RETURN                         2318

                          TOTAL  =           2318
```

Observe que os dois programas apresentados são fundamentalmente diferentes. O primeiro apenas interrompe o processamento normal do programa até que a barra de espaços seja pressionada, enquanto o segundo desvia o processamento para uma sub-rotina, esteja ele em que linha estiver, sempre que a barra de espaços for pressionada.

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 112, 159, 160, 178 e 179.
Coleção de Programas para MSX v.2 - páginas 75 a 82.

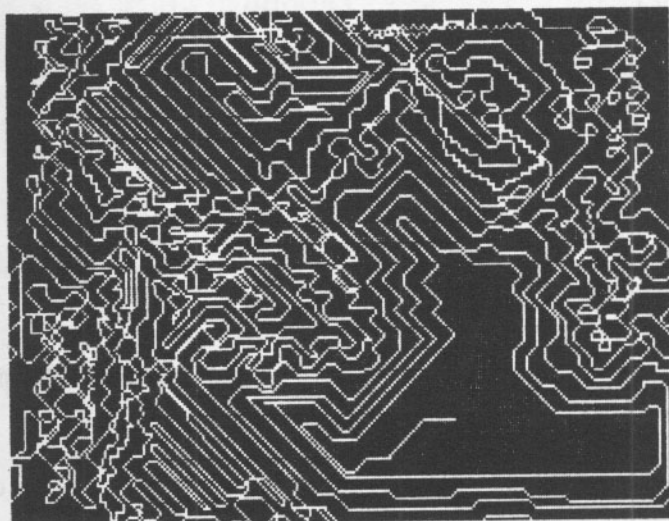
1.A - USANDO AS TECLAS DE SETAS

As teclas de setas podem ser testadas através de uma função do BASIC MSX. O programa apresentado a seguir ilustra um uso típico dessa função, associada ao controle de um ponto plotante na SCREEN 2.

Note que usamos "parênteses lógicos" para tornar a tela ilimitada. Se o ponto tentar sair da tela por qualquer um de seus 4 lados, automaticamente será transportado para a primeira posição do lado oposto.

```
100 SCREEN 2,,0 : DEFINT A-Z                     4BC
110 X=128 : Y=96                                 99C
120 PSET (X,Y),15                                C74
130 A=STICK(0) : IF A=0 THEN 130                 14F6
140 Y=Y+(A<3 OR A=8)-(A>3 AND A<7)               2178
150 X=X-(A>1 AND A<5)+(A>5)                      2ADA
160 X = -255*(X=-1) - X*(X<>-1)                  3A22
170 X = -X*(X<>256)                              43CB
180 Y = -191*(Y=-1) - Y*(Y<>-1)                  51D3
190 Y = -Y*(Y<>192)                              59B4
200 GOTO 120                                     5E2E

                              TOTAL =            5E2E
```

Para entender o programa, lembre-se que quando o conteúdo dentro dos parênteses for verdadeiro, ele pode ser substituído pelo valor numérico "-1", e quando for falso, pelo valor "0".

1.8 - USO DA INSTRUÇÃO ON KEY GOSUB

As teclas de funções podem ser usadas para chamar sub-rotinas automaticamente durante as interrupções. Observe o programa a seguir. Digite-o e rode-o.

```
10 SCREEN 0,,0 : PLAY "L64"                          3E1
20 FOR F=1 TO 10 : KEY(F) ON : NEXT F               B37
30 ON KEY GOSUB 100,200,300,400,500,600,           1CE6
700,800,900,1000
40 GOTO 40                                          1FEB
100 PRINT "  F1 PRESSIONADA !!! "                   27AC
110 PLAY "C"                                        2B2E
120 RETURN                                          2CB9
200 PRINT "  F2 PRESSIONADA !!! "                   3EEC
210 PLAY "C#"                                       437B
220 RETURN                                          44B8
300 PRINT "  F3 PRESSIONADA !!! "                   5875
310 PLAY "D"                                        5D9E
320 RETURN                                          5F5F
400 PRINT "  F4 PRESSIONADA !!! "                   72FE
410 PLAY "D#"                                       7891
420 RETURN                                          7A54
500 PRINT "  F5 PRESSIONADA !!! "                   839A
510 PLAY "E"                                        867D
520 RETURN                                          8740
600 PRINT "  F6 PRESSIONADA !!! "                   8E5F
610 PLAY "F"                                        9101
620 RETURN                                          923B
700 PRINT "  F7 PRESSIONADA !!! "                   9AB4
710 PLAY "F#"                                       9D7E
720 RETURN                                          9F86
800 PRINT "  F8 PRESSIONADA !!! "                   A8E9
810 PLAY "G"                                        AD2C
820 RETURN                                          AE72
900 PRINT "  F9 PRESSIONADA !!! "                   C083
910 PLAY "G#"                                       C4E3
920 RETURN                                          C5F7
1000 PRINT " F10 PRESSIONADA !!! "                  D798
1010 PLAY "A"                                       DC10
1020 RETURN                                         D061

                              TOTAL    =           D061
```

Observe que, independentemente da linha do programa que estiver sendo executada, sempre que alguma tecla de função for pressionada uma sub-rotina será chamada.

DICAS PARA USAR O VÍDEO

Este capítulo aborda os recursos do vídeo do MSX. Existe um circuito dedicado ao controle do vídeo (VDP) que tem à sua disposição 16 Kbytes de memória RAM (VRAM) para armazenar os dados da tela. O controle do vídeo pode ser feito através do BASIC, com os vários comandos dedicados a isso, ou diretamente em Linguagem de Máquina.

2.1 - Cor de frente igual a cor de fundo 22
2.2 - Pseudo-borda na SCREEN 0 23
2.3 - Textos na SCREEN 2 24
2.4 - Caracteres menores que 32 26
2.5 - Caracteres de controle do vídeo 27
2.6 - Usando o STEP em comandos gráficos 29
2.7 - Posicionando o cursor na SCREEN 2 31
2.8 - Movimentos na tela 32
2.9 - SCREEN 4 35
2.A - Setores com o CIRCLE 39
2.B - Redefinindo caracteres 40
2.C - SCROLL UP para SCREEN 0 52
2.D - SCROLL DOWN para SCREEN 0 53
2.E - SCROLL LEFT para SCREEN 0 54
2.F - SCROLL RIGHT para SCREEN 0 55
2.G - SCROLL UP para SCREEN 1 56
2.H - SCROLL DOWN para SCREEN 1 57
2.I - SCROLL LEFT para SCREEN 1 58
2.J - SCROLL RIGHT para SCREEN 1 59
2.K - Centralizando caracteres 60
2.L - Animação com SPRITES 61
2.M - Letras ampliadas 63
2.N - Entendendo o DRAW 66
2.O - "WARP 8" na SCREEN 2 67
2.P - "SPRITEANDO" a tabela de caracteres ... 68
2.Q - Arlequim bêbado 69
2.R - Usando 40 ou 64 colunas na SCREEN 2 ... 71
2.S - Carimbador de SPRITES 8x8 na SCREEN 2 . 73
2.T - Carimbador de SPRITES 16x16 na SCREEN 2 75
2.U - Armazenando telas na RAM 76
2.V - Usando a VRAM para dados 77
2.W - Pesquisador de desenhos 79
2.X - Imagens instantâneas 83
2.Y - Impressão em tamanho duplo na SCREEN 2 84

2.1 - COR DE FRENTE IGUAL A COR DE FUNDO

Muitos programas ao terminarem a execução ou serem interrompidos por CONTROL + STOP deixam a tela com a cor de frente igual a cor de fundo e a primeira impressão que se tem é que o micro quebrou. Digite e rode o programinha listado a seguir:

```
10 SCREEN 2                  100
20 CIRCLE (128,86),50        3E4
30 COLOR 1                   4EF
40 GOTO 40                   670
```

Após o desenho do círculo terminar, digite CONTROL + STOP. A tela deverá ficar totalmente escura.

Para verificar o que está acontecendo basta digitar:

SHIFT + HOME/CLS e
SHIFT + F1

Com isso, a tela será limpa e as cores normais do vídeo serão restabelecidas, desde que as teclas de funções não tenham sido redefinidas pelo programa que foi interrompido.

Se o procedimento descrito acima não funcionar, tente digitar o comando abaixo, mesmo sem vê-lo na tela:

COLOR 15,1 : SCREEN 0 (e RETURN)

2.2 - PSEUDO-BORDA NA SCREEN 0

O comando COLOR, quando usado com a SCREEN 0, não permite a especificação da cor da borda. Com um pequeno programa em BASIC podemos resolver parcialmente o problema, gerando uma PSEUDO-BORDA para a SCREEN 0. Digite e execute o programa a seguir e depois verifique as novas características da tela.

```
10 COLOR 1,15 : SCREEN 0                          208
20 FOR F=2048 TO 4095                             784
30   X = NOT(VPEEK(F)) AND 255                    DC8
40   VPOKE F, X                                   105F
50 NEXT F                                         11CF
60 INPUT"Qual a cor da borda (0-15)";B            13F6
70 IF B<0 OR B>15 THEN 60                         2420
80 COLOR ,B                                       267F
90 GOTO 60                                        2A8A
```

TOTAL = 2A8A

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.3 - TEXTOS NA SCREEN 2

O MSX permite a impressão de letras e gráficos na SCREEN 2. Para isso é necessário abrir um arquivo na tela (GRP:) e usar a instrução PRINT #. Observe o programa a seguir:

```
10 SCREEN 2                              100
20 OPEN "GRP:" AS #1                     409
30 CIRCLE (128,86),50                    786
40 PRESET (110,84)                       987
50 PRINT #1, "EDITORA ALEPH"             F27
60 GOTO 60                              1219

                          TOTAL  =      1219
```

A linha 10 seleciona o uso da tela gráfica de alta resolução (GRP:).

A linha 20 abre um arquivo nessa tela. Não é necessário especificar o tipo de arquivo (...FOR OUTPUT...), uma vez que ele só pode ser de saída!

A linha 30 desenha um círculo de centro na posição (128,86) e raio de 50 pixels.

A linha 40 "marca" um ponto na posição (110,84), a partir do qual a mensagem será impressa.

A linha 50 imprime a mensagem na tela a partir do ponto marcado pela linha 40. Cada caractere é definido dentro de uma matriz de 8x8 pontos. A posição marcada pela linha 40 posiciona o vértice superior esquerdo do primeiro caractere da mensagem.

Observe que podemos "criar" novos tipos de letras na SCREEN 2, usando uma dupla impressão dos caracteres normais. Experimente inserir no programa anterior as seguinte linhas:

```
55 PRESET (109,84)
56 PRINT #1, "EDITORA ALEPH"
```

Isso deve ter produzido uma mensagem em "bold"!

Um outro recurso é o uso de espaçamento menor entre as letras a serem impressas. Experimente executar o programa a seguir. Ele imprime na SCREEN 2 uma mensagem com espaçamento reduzido.

```
10 SCREEN 2                              100
20 OPEN "GRP:" AS #1                     409
30 A$="EDITORA ALEPH"                    B5B
```

```
40 FOR F=1 TO LEN(A$)
50   PRESET (110+(F-1)*6,84)
60   PRINT #1,MID$(A$,F,1)
70 NEXT F
80 GOTO 80
```

TOTAL = 2489

Para entender melhor o funcionameneto do programa, experimente substituir a linha 50 por:

```
50  PRESET (110+(F-1)*12,84)
```

BIBLIOGRAFIA RECOMENDADA:

Curso de BASIC v.1 - páginas 65 e 66.
Coleção de Programas para MSX - páginas 32, 33, 59 e 60.

2.4 - CARACTERES MENORES QUE 32

Os micros MSX dispõem de 256 caracteres, todos apresentáveis no vídeo. Entretanto, os primeiros 32 caracteres correspondem a códigos de controle de periféricos (0 a 31) e para serem apresentados através de seus códigos necessitam de uma sintaxe peculiar da instrução PRINT CHR$. O programa a seguir apresenta os 32 caracteres de controle através de seus códigos.

```
10 SCREEN 1                          10E
20 WIDTH 16                          218
30 FOR F=0 TO 31                     59F
40  PRINT CHR$(1)+CHR$(64+F);        01C
50 NEXT F                            E99

                        TOTAL  =     E99
```

Note que a apresentação é feita pela linha 40. Para se mostrar um caractere de controle através de seu código é necessário usar a sintaxe:

```
PRINT CHR$(1)+CHR$(64+ nº )
```

Onde "nº" é o código (de 0 a 31) do caractere a ser apresentado.

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.1 - páginas 26 e 27.

2.5 - CARACTERES DE CONTROLE DE VÍDEO

O padrão de vídeo dos MSX é um dos mais utilizados em microcomputadores. Alguns caracteres podem ser usados para controlar o vídeo em funções como o posicionamento do cursor, limpeza da tela, formato do cursor, etc.

A seguir relacionamos os caracteres de controle do vídeo do MSX. Eles devem ser usados com o comando PRINT. Por exempo, para deixar o cursor posicionado na linha 10 e coluna 20 basta comandar:

```
PRINT CHR$(27)CHR$(32+10)CHR$(32+20);
```

Apaga a tela
CHR$(27)"j" **ou**
CHR$(27)"E"

Apaga do cursor ao fim da linha
CHR$(27)"K"

Apaga do cursor ao fim da tela
CHR$(27)"J"

Apaga a linha inteira
CHR$(27)"l"

Insere uma linha em branco
CHR$(27)"L"

Elimina uma linha
CHR$(27)"M"

Posiciona cursor
CHR$(27)"Y";
CHR$(32+nº da linha);
CHR$(32+nº da coluna)

Cursor linha acima
CHR$(27)"A"

Cursor linha abaixo
CHR$(27)"B"

Cursor coluna a direita
CHR$(27)"C"

Cursor coluna a esquerda
CHR$(27)"D"

Cursor em HOME
CHR$(27)"H"

Cursor inteiro
CHR$(27)"x4"

Cursor pela metade
CHR$(27)"y4"

Cursor apagado
CHR$(27)"x5"

Cursor aceso
CHR$(27)"y5"

O programa a seguir ilustra o uso de alguns dos recursos descritos acima. Digite-o, rode-o e estude-o.

```
10 SCREEN 0 : WIDTH 38 : KEY OFF                  0220
11 PRINT "Digite qualquer coisa ";               1056
12 PRINT "e use as teclas de setas";             1506
13 PRINT " e as teclas HOME e CLS !"             105A
14 PRINT : PRINT : PRINT                         22BC
20 A$=INKEY$ : IF A$="" THEN 20                  201A
30 IF A$<>CHR$(12) THEN 50                       9768
40    PRINT CHR$(27)"j"                          9F03
50 IF A$<>CHR$(11) THEN 70                       4A80
60    PRINT CHR$(27)"K"                          36F6
70 PRINT A$;                                     3C19
90 GOTO 20                                       5051

                                   TOTAL  =      6051
```

BIBLIOGRAFIA RECOMENDADA:

Sistema de Disco para MSX - página 106.

2.6 - USANDO O STEP EM COMANDOS GRÁFICOS

A instrução auxiliar STEP pode ser muito útil quando usada com comandos gráficos.

Veja o programa a seguir:

```
10 SCREEN 2
20 PSET (20,20)
30 PSET STEP (20,20)
40 GOTO 40
```

TOTAL = 185

Ao ser executado ele irá plotar dois pontos na diagonal que sai do canto superior esquerdo da tela.

Isso acontece porque a instrução STEP permite a definição de um novo sistema de coordenadas na tela.

O sistema de coordenadas normal é um sistema ABSOLUTO, isto é, suas coordenadas são sempre correspondentes a uma mesma orígem, fixa no canto superior esquerdo da tela.

O sistema usado pelo STEP é um sistema MóVEL, em que as coordenadas correspondem a uma orígem móvel, definida pelo último ponto "marcado" na tela. Note que PONTO "MARCADO" não é necessariamente PONTO "PLOTADO". Por exemplo, ao se usar um comando CIRCLE o ponto "marcado" é o seu centro, enquanto que os pontos plotados são os do círculo.

Vamos tentar entender isso melhor.

Observe novamente o programa anterior.

A linha 10 apenas seleciona a SCREEN 2 do micro. Essa tela tem uma resolução de 256 colunas x 192 linhas, num total de 49152 pontos.

A linha 20 marca o ponto de coordenadas x=20 e y=20 na tela. Note que essas coordenadas são referentes ao sistema ABSOLUTO, pois NÃO existe a instrução STEP precedendo as coordenadas.

A linha 30 também marca um ponto na tela (o de coordenadas x'=20 e y'=20), porém as coordenadas desse ponto são precedidas pela instrução STEP e, portanto, referem-se ao sistema de coordenadas MóVEL. Como o último ponto marcado na tela foi o de coordenadas ABSOLUTAS x=20 e y=20, esse ponto foi tomado como ORíGEM do sistema MóVEL. Portanto, as coordenadas x'=20 e y'=20 do sistema MóVEL correspondem às coordenadas x=40 e y=40 do sistema ABSOLUTO.

Agora, esquente um pouco a cabeça tentando entender o funcionamento dos três programinhas listados na próxima página.

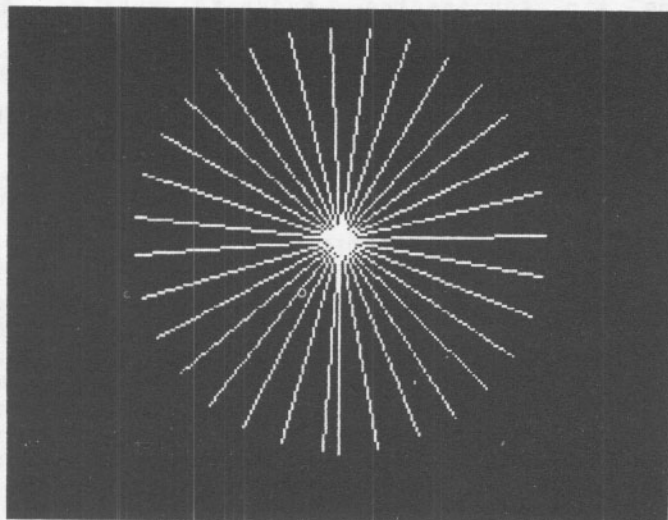
PROGRAMA 1

```
10 SCREEN 2                          10D
20 PSET (0,0)                        288
30 CIRCLE STEP(6,6),4                574
40 GOTO 30                           705
```

PROGRAMA 2

```
10 SCREEN 2                          10D
20 PSET (0,0)                        288
30 LINE STEP(3,3)-STEP(3,3)          7A0
40 GOTO 30                           8D4
```

PROGRAMA 3

```
10 SCREEN 2                          10D
20 FOR F=0 TO 6.28 STEP .2           7D2
30   X=80*SIN(F) : Y=80*COS(F)       11DC
40   LINE STEP(X,Y)-(128,86)         1868
50 NEXT F                            1A85
60 GOTO 60                           1DAE
```

2.7 - POSICIONANDO O CURSOR NA SCREEN 2

A forma mais genérica de se posicionar o cursor na SCREEN 2 é usando o seguinte comando:

```
PSET (coluna,linha),POINT (coluna,linha)
```

Por exemplo, para posicionar o cursor na coluna 89 e linha 190 devemos fazer:

```
PSET (89,190),POINT (89,190)
```

Se quisermos posicionar o cursor para impressão de textos na SCREEN 2 podemos considerá-la dividida em 32 colunas e 24 linhas, como a SCREEN 1. Desse modo, o comando PSET acima pode ser substituido por:

```
PSET (8*col,8*lin),POINT (8*col,8*lin)
```

Para imprimir na coluna 10 e linha 5 devemos usar então:

```
PSET (8*10,8*5),POINT (8*10,8*5)
```

Agora, digite e rode o programa a seguir:

```
10 SCREEN 2                                  100
20 OPEN "GRP:" AS #1                         4C9
30 C = 10 : L = 10                           8CE
40 PSET(8*C,8*L),POINT(8*C,8*L)              1108
50 PRINT #1, "ALEPH"                         150C
60 GOTO 50                                   1813

                                   TOTAL  =  1813
```

Na linha 30 define-se a coluna (de 0 a 31) e a linha (de 0 a 23) onde desejamos imprimir. A linha 40 posiciona o cursor e a linha 50 imprime na tela.

Veja a dica 2.3 para entender melhor o programa.

2.8 - MOVIMENTOS NA TELA

Muitas vezes pode-se desejar que a tela do MSX apresente algum movimento global. A seguir, apresentamos seis programas que produzem movimentos globais na SCREEN 1.

PROGRAMA 1

```
10 ' TESTE DE MOVIMENTO                          763
20 SCREEN 1 : WIDTH 32 : COLOR 1,7,4             88E
30 FOR F=0 TO 22                                 F56
40 PRINT STRING$(32,"O"); : NEXT F              170F
50 FOR F=8*ASC("O") TO 8*ASC("O")+7             2471
60 IF B=0 THEN VPOKE F,VPEEK(F)/8               20F7
70 IF B=1 THEN VPOKE F,8*VPEEK(F)               3976
80 NEXT F : B=(B+1) MOD 2 : GOTO 50             4950

                               TOTAL   =        4950
```

PROGRAMA 2

```
100 ' TESTE DE MOVIMENTO 2                       81E
110 SCREEN 1 : WIDTH 32 : COLOR 1,7,4            C07
120 FOR F=0 TO 22                               1029
130 PRINT STRING$(32,"."); : NEXT F             1901
140 A=VPEEK(8*ASC("."))                         21C3
150 FOR F=8*ASC(".") TO 8*ASC(".")+6            2F53
160 VPOKE(F),VPEEK(F+1)                         377E
170 NEXT F                                      39C6
180 VPOKE F,A                                   3DEA
190 GOTO 140                                    40B0

                               TOTAL   =        40B0
```

PROGRAMA 3

```
200 SCREEN1 : WIDTH 32                           330
210 FOR F=48 TO 69                               815
220 A$=A$+CHR$(F)                                DEE
230 NEXT F                                       FF5
240 FOR F=1 TO 31                               1497
250 A=INT(1+RND(1)*21)                          1D07
260 A$=RIGHT$(A$,A)+LEFT$(A$,21-A)              2865
270 FOR G=1 TO 21:LOCATE F,G                    2FC7
280 PRINT MID$(A$,G,1):NEXT G                   387F
290 NEXT F                                      3B29
300 FOR F=48*8 TO 69*8+7                        459A
310 A=INT(8*RND(1))                             4CE7
```

```
320 IF RND(1)>.01 THEN VPOKE F,0 ELSE VP 61C5
OKE F,2^A
330 NEXT F                                64F4
340 A%=VPEEK(69*8+7)                      6D96
350 FOR F%=69*8+6 TO 48*8 STEP-1          7C03
360 VPOKE F%+1,VPEEK(F%)                  830D
370 NEXT F%                               84BD
380 VPOKE F%+1,A%                         8879
390 GOTO 340                              8AAA

                         TOTAL   =        8AAA
```

PROGRAMA 4

```
400 SCREEN1 : WIDTH 32                    2F9
410 FOR F=48 TO 69                        795
420 A$=A$+CHR$(F)                         D67
430 NEXT F                                F85
440 FOR F=1 TO 31                         1390
450 A=INT(1+RND(1)*21)                    1CF9
460 A$=RIGHT$(A$,A)+LEFT$(A$,21-A)        27BF
470 FOR G=1 TO 21:LOCATE F,G              2F69
480 PRINT MID$(A$,G,1):NEXT G             3849
490 NEXT F                                3ABB
500 FOR F=48*8 TO 69*8+7                  4504
510 A=INT(8*RND(1))                       4C09
520 IF RND(1)>.01 THEN VPOKE F,0 ELSE VP 609C
OKE F,2^A
530 NEXT F                                6424
540 A%=VPEEK(48*8)                        6C5A
550 FOR F%=48*8 TO 69*8+6                 7894
560 VPOKE F%,VPEEK(F%+1)                  8111
570 NEXT F%                               827C
580 VPOKE F%,A%                           84FF
590 GOTO 540                              867C

                         TOTAL   =        867C
```

PROGRAMA 5

```
600 ' PSEUDO-SCROLL A ESQUERDA                   991
610 SCREEN 1 : WIDTH 32 : COLOR 1,7,4           DA8
620 X=8*ASC(" ")                                12EB
630 B=0 : VPOKE X,&B00000001                    1AA5
640 VPOKE X,VPEEK(X)*2                          209F
650 B=B+1 : IF B=7 THEN 630                     2998
660 GOTO 640                                    2E42

                            TOTAL   =           2E42
```

PROGRAMA 6

```
670 ' PSEUDO-SCROLL A DIREITA                    983
680 SCREEN 1 : WIDTH 32 : COLOR 1,7,4           DDA
690 X=8*ASC(" ")                                1312
700 B=0 : VPOKE X,&B10000000                    1BB0
710 VPOKE X,VPEEK(X)/2                          218C
720 B=B+1 : IF B=7 THEN 700                     2AFC
730 GOTO 710                                    2F0D

                            TOTAL   =           2F0D
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.9 - SCREEN 4

Quando comandamos SCREEN 1 ou SCREEN 2, o interpretador BASIC prepara a VRAM com as tabela próprias para a tela selecionada e o VDP para agir sobre essas tabelas.

Analisando os procedimentos executados pelo micro quando usamos um comando SCREEN podemos idealizar uma forma de fazer a VRAM ser preparada com as tabelas da SCREEN 2 e as variáveis do sistema serem preparadas para operar sobre a SCREEN 1! Desse modo, poderemos conciliar a rapidez de operação da SCREEN 1 com todos os recursos de cores e formas da SCREEN 2.

A idéia é "enganar" o interpretador, fazendo-o pensar que está operando sobre a SCREEN 1, enquanto a VRAM e o VDP estarão preparados para a SCREEN 2. Note que as variáveis do sistema, preparadas pelo interpretador para uma dada SCREEN, funcionam como um "mecanismo inibitório" para o VDP, fazendo-o comportar-se sob rígido controle. Se usarmos diretamente as rotinas do BIOS, sem que o interpretador "saiba", poderemos evitar os "mecanismos inibitórios" e gerar algo parecido com uma "ESQUIZOFRENIA INDUZIDA" no micro.

Observe atentamente o programa a seguir. Digite-o e, antes de executá-lo, leia os comentários que o seguem.

PROGRAMA SCREEN 4

```
100 COLOR 15,1,15                              1CF
110 SCREEN 2                                   33B
120 DEFINT A-Z                                 62E
130 SCREEN 1                                   809
140 DEFUSR = &H7E                              3B7
150 X = USR(0)                                 F28
160 FOR F = 0 TO 2047                          1483
170   X = PEEK (&H1BBF + F)                    10FC
180   VPOKE (         F),X                     2615
190   VPOKE (&H800  + F),X                     2CF4
200   VPOKE (&H1000 + F),X                     3602
210 NEXT F                                     393C
220 FOR F=0 TO 7                               3EE2
230 VPOKE 2048+255*8+F,255                     4811
240 VPOKE 4096+255*8+F,255                     511F
250 NEXT F                                     53C7

                                   TOTAL  =    53C7
```

Antes de mais nada, lembre-se de como a VRAM fica dividida quando se usa a SCREEN 1 ou a SCREEN 2.

Uma vez estudada a estrutura da VRAM quando no modo SCREEN 1 ou SCREEN 2, podemos analisar o programa.

A linha 100 serve apenas para que a cor da borda da tela seja selecionada quando o comando SCREEN na linha 110 for executado.

A linha 110 seleciona a SCREEN 2 para que as tabelas da VRAM sejam preparadas para ela.

A linha 120 serve apenas para aumentar um pouco a velocidade de execução do programa, afinal ele necessita apenas de valores inteiros. Este recurso dos micros MSX é extremamente útil para esta finalidade.

A linha 130 seleciona a SCREEN 1 para que as variáveis do sistema indiquem ao interpretador o modo SCREEN 1. Note que até aqui não houve nenhum "truque" de programação digno de maiores explicações. O interpretador ainda não foi "enganado"! Isso só ocorrerá após a execução das próximas linhas.

As linhas 140 e 150 executam a rotina do BIOS SETGRP (em &H007E). Ela prepara o VDP para acessar as tabelas da SCREEN 2. Note que, com isso, já temos a VRAM preparada como SCREEN 1 (e residualmente como SCREEN 2) e o VDP preparado para acessar a SCREEN 2.

As linhas de 160 a 210 carregam as tabelas de caracteres da VRAM com os caracteres da ROM (de 0 a 255). Os desenhos dos 255 caracteres são definidos três vezes, uma para cada terço da tela. A tabela correspondente às linhas de 0 a 7 da tela ocupa os primeiros 2 Kbytes da VRAM. De 2 a 4 Kbytes está a tabela correspondente às linhas de 8 a 15. As linhas de 16 a 23 usam a tabela de caracteres entre 4 e 6 Kbytes.

As linhas de 220 a 250 apenas redefinem os desenhos do caractere 255 (cursor) para serem usados no segundo e terceiro terço da tela.

Note que a linha 180 poderia ter sido omitida sem nenhum problema, pois o comando SCREEN 1 da linha 130 já havia carregado os desenhos dos 255 caracteres no primeiro terço da tela.

ALTERANDO AS CORES DOS CARACTERES

Na SCREEN 1 podemos alterar a cor de grupos de 8 caracteres. Por exemplo, se alterarmos a cor do caractere 3, as cores dos caracteres de 0 a 7 serão simultaneamente alteradas. Na SCREEN 2, podemos alterar a cor de cada um dos caracteres

individualmente. Podemos ainda ir mais além e definir, para cada caractere, 16 diferentes cores! Isso mesmo, 16 cores em cada caractere!

Acrescente ao programa anterior as linhas mostradas a seguir e rode-o novamente. Com isso, os caracteres dos números e dos parênteses terão suas cores redefinidas no 1º terço da tela!

```
260 X=8192+8*ASC("(")                  5F25
270 FOR F=X TO X+7                     66E0
280   VPOKE F,&B1000001                73F5
290 NEXT F                             77B2
300 X=8192+8*ASC(")")                  8055
310 FOR F=X TO X+7                     8534
320   VPOKE F,&B1000001                8905
330 NEXT F                             8AB6
340 X=8192+8*ASC("0")                  9084
350 Y=8192+8*ASC("9")+7                9995
360 FOR F=X TO Y                       9DA4
370   VPOKE F,&B11010001               A3BC
380 NEXT F                             A5A5

                           TOTAL   =   A5A5
```

Agora vamos fazer com que o cursor seja redefinido com as 16 cores. Acrescente também ao programa as linhas mostradas a seguir. Depois, execute-o.

```
390 F=8192+8*255                       A878
400 VPOKE F+0,&B00001000               B7A5
410 VPOKE F+1,&B00011001               C2A3
420 VPOKE F+2,&B00101010               CCCC
430 VPOKE F+3,&B00111011               D8AA
440 VPOKE F+4,&B01001100               E621
450 VPOKE F+5,&B01011101               F201
460 VPOKE F+6,&B01101110               FF0C
470 VPOKE F+7,&B01111111               474
480 F=10240+8*255                      9C1
490 VPOKE F+0,&B00001000               E80
500 VPOKE F+1,&B00011001               12C4
510 VPOKE F+2,&B00101010               1A87
520 VPOKE F+3,&B00111011               2172
530 VPOKE F+4,&B01001100               27C6
540 VPOKE F+5,&B01011101               2FB3
550 VPOKE F+6,&B01101110               3B13
560 VPOKE F+7,&B01111111               4610
570 F=12288+8*255                      4CCA
580 VPOKE F+0,&B00001000               5A66
```

```
590 VPOKE F+1,&B00011001      71BF
600 VPOKE F+2,&B00101010      74CF
610 VPOKE F+3,&B00111011      81B0
620 VPOKE F+4,&B01001100      8706
630 VPOKE F+5,&B01011101      8C81
640 VPOKE F+6,&B01101110      9DF3
650 VPOKE F+7,&B01111111      961E

                  TOTAL =     961E
```

Agora, o cursor está colorido!

Os exemplos apresentados são bem simples para facilitar a compreensão. Os recursos oferecidos pela SCREEN híbrida que apresentamos são, entretanto, muito mais vastos.

Você deve ter percebido que o programa demora vários segundos para ser executado. Isso é aceitável quando levamos em conta que o BASIC tem que acessar quase 16 Kbytes, entretanto para os programadores mais exigentes a demora pode ser um fator muito negativo.

Podemos pensar, então, em transformar o programa numa rotina em Linguagem de Máquina. Indo além, podemos pensar numa rotina que permita a implementação do comando SCREEN 4 no BASIC, de modo que ao ser executado ele gere a tela híbrida com a mesma velocidade que as outras SCREEN's. Para facilitar a alteração das cores dos caracteres podemos imaginar um novo comando do BASIC ou ainda o aproveitamento de comandos não implementados como o IPL ou o CMD. Isto, entretando, já é assunto para um texto mais extenso.

Se você não quiser esperar, poderá encontrar estas e muitas outras idéias já executadas e analisadas no livro PROGRAMAÇÃO AVANÇADA EM MSX. Para estudar mais detalhadamente a estrutura da VRAM nas várias SCREEN's, as rotinas do BIOS e as Variáveis do Sistema, sugerimos a leitura atenta do livro APROFUNDANDO-SE NO MSX. Nesses dois livros os assuntos são tratados de forma bastante completa.

Exemplos e aplicações práticas comentadas passo a passo podem ser encontradas nos livros COLEÇÃO DE PROGRAMAS PARA MSX, volumes 1 e 2.

Para completar seu conhecimento sobre a SCREEN 1, veja a dica 2.Q (Arlequim Bêbado).

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.A - SETORES COM CIRCLE

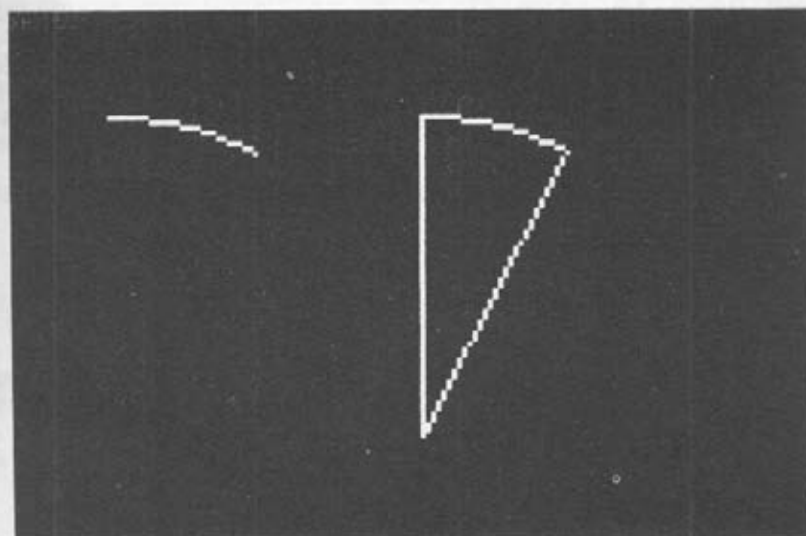
O comando CIRCLE do BASIC permite o traçado de arcos de circunferências e de perímetros de setores circulares.

Para traçar setores, basta usar valores negativos para os ângulos inicial e final. Veja o programa a seguir:

```
10 SCREEN 2                                    100
20 PI = 4*ATN(1)                               60E
30 CIRCLE (70,80),60,15,PI/3,PI/2,1            1089
40 CIRCLE (130,80),60,15,-PI/3,-PI/2,1         1690
50 GOTO 50                                     1948
```

A linha 20 define a variável PI, atribuindo-lhe o valor da constante matemática π.

A linha 30 traça um arco de circunferência normal.

A linha 40 traça o perímetro de um setor, pois os ângulos são negativos.

2.8 - REDEFININDO CARACTERES

Uma maneira rápida de redefinir caracteres é acessar a tabela de definição diretamente na VRAM. Essa tabela ocupa posições diferentes em cada tipo de tela (SCREEN's 0, 1, 2 ou 3). A seguir apresentamos 3 programas bem curtos que redefinem, apenas como exemplo, o formato do caractere de código 65 (ou &H41), originalmente a letra "A" nas SCREEN's 0, 1 e 2.

PARA A SCREEN 0

```
100 SCREEN 0
110 E=2048+8*65
120 FOR F=E TO E+7
130  READ A$
140  VPOKE F,VAL("&B"+A$)
150 NEXT F
160 PRINT : PRINT "CHR$(65)=";CHR$(65)
170 PRINT : LIST
180 DATA 00111100
190 DATA 01000010
200 DATA 10011001
210 DATA 10111101
220 DATA 10111101
230 DATA 10011001
240 DATA 01000010
250 DATA 00111100
```

PARA A SCREEN 1

Apenas substitua as linhas 100 e 110 do programa anterior por:

```
100 SCREEN 1
110 E=8*65
```

PARA A SCREEN 2

```
100 SCREEN 2
110 FOR F=6144 TO 6144+24*32-1
120   VPOKE F,65
130 NEXT F
140 FOR F=0 TO 4096 STEP 2048
```

```
150    FOR G=F+8*65 TO F+8*65+7                  106A
160      READ A$ : VPOKE G,VAL("&B"+A$)          28DD
170      VPOKE G+8192,&B00011111                 368A
180    NEXT G                                    3A75
190    RESTORE                                   3CBC
200 NEXT F                                       3F53
210 GOTO 210                                     41C4
220 DATA 00111100                                4866
230 DATA 01000010                                4E72
240 DATA 10011001                                57F0
250 DATA 10111101                                61A0
260 DATA 10111101                                6BE9
270 DATA 10011001                                759C
280 DATA 01000010                                7ED1
290 DATA 00111100                                85F8

                              TOTAL   =          85F8
```

Essa forma de redefinir caracteres, entretanto, nem sempre é a melhor, pois quando um comando SCREEN é executado as tabelas da VRAM são automaticamente recarregadas. Para entender melhor esse problema, rode novamente o programa feito para a SCREEN 0 e depois comande SCREEN 0. Você verá que o caractere de código 65 voltou a ser a letra "A". A redefinição feita na VRAM foi destruída pelo comando SCREEN.

Para contornar esse problema podemos redefinir toda a tabela de caracteres na própria RAM e fazer com que o comando SCREEN carregue essa nova tabela para a VRAM, ao invés de usar a tabela da ROM. O programa apresentado a seguir se presta a facilitar a redefinição dos caracteres de forma mais permanente. Digite e grave o programa. Depois veremos como usá-lo.

```
1000 KEYOFF : SCREEN 0 : WIDTH 40               604
1010 FOR E=&HB000 TO &HBAEF                     C8A
1020   READ C$ : POKE E,VAL("&H"+C$)            1300
1030 NEXT E                                     15AB
1040 DEFUSR=&HB000 : X=USR(0)                   10A9
1050 DATA 3E,01,21,AB,FC,77,CD,1E               2642
1060 DATA B1,CD,E5,B0,CD,26,B2,21               3153
1070 DATA 08,00,22,B7,FC,21,B8,00               4380
1080 DATA 22,B9,FC,21,CB,BA,7E,2F               5504
1090 DATA FE,00,28,06,CD,8D,00,23               6C17
1100 DATA 18,F4,16,08,CD,57,B2,FE               8138
1110 DATA 53,20,06,AF,21,AB,FC,77               8E6A
1120 DATA C9,21,09,B0,E5,FE,43,CA               9483
1130 DATA 96,B2,FE,0D,28,1F,0E,01               9A20
```

```
1140 DATA FE,1C,28,11,0E,FF,FE,1D     A297
1150 DATA 28,0B,0E,F0,FE,1E,28,05     AAA0
1160 DATA 0E,10,FE,1F,C0,3A,C9,B2     B7D4
1170 DATA B1,32,C9,B2,C9,CD,0E,B2     CE72
1180 DATA 16,02,CD,57,B2,FE,0D,C8     DF3B
1190 DATA 21,65,B0,E5,01,00,FE,FE     F58C
1200 DATA 20,28,24,0C,FE,4D,28,1F     9AE
1210 DATA FE,1C,28,11,0E,FF,FE,1D     10BE
1220 DATA 28,0B,0E,F8,FE,1E,28,05     15F6
1230 DATA 0E,08,FE,1F,C0,3A,CA,B2     1D30
1240 DATA B1,E6,3F,32,CA,B2,C9,CD     2606
1250 DATA 46,B2,3A,CA,B2,F5,0F,0F     3229
1260 DATA 0F,E6,07,5F,16,00,FD,19     4302
1270 DATA F1,E6,07,3C,CB,08,CB,09     5506
1280 DATA 3D,20,F9,FD,7E,00,A0,B1     6A68
1290 DATA FD,77,00,CD,E5,B0,CD,46     7EFB
1300 DATA B2,CD,26,B2,CD,CB,B1,06     8B8F
1310 DATA 08,D5,E5,3E,08,FD,5E,00     90FD
1320 DATA CD,EC,B1,E1,D1,CD,E0,B1     97FB
1330 DATA FD,23,10,ED,C9,CD,46,B2     A144
1340 DATA 0E,BF,1E,07,CD,CB,B1,06     A92D
1350 DATA 08,0E,05,C5,D5,E5,06,08     B80D
1360 DATA FD,7E,00,07,F5,9F,5F,3E     CCCB
1370 DATA 05,CD,EC,B1,CD,D6,B1,CD     E03C
1380 DATA D6,B1,F1,10,EE,E1,D1,C1     F518
1390 DATA CD,E0,B1,0D,20,DD,CD,E0     850
1400 DATA B1,FD,23,10,D4,C9,01,00     E62
1410 DATA 08,11,CB,B2,2A,20,F9,C5     133C
1420 DATA D5,3A,1F,F9,CD,0C,00,FB     1AE7
1430 DATA D1,C1,12,13,23,0B,78,B1     2495
1440 DATA 20,ED,CD,72,00,3A,E9,F3     3173
1450 DATA 07,07,07,07,4F,3A,EA,F3     432F
1460 DATA B1,01,00,18,2A,C9,F3,CD     53F5
1470 DATA 56,00,21,0B,B1,01,0A,FF     69A9
1480 DATA 1E,06,3E,11,CD,8A,B1,21     7DA1
1490 DATA 06,31,01,BE,AA,1E,06,3E     893B
1500 DATA 09,CD,8A,B1,21,30,31,01     8E7D
1510 DATA BE,FF,1E,06,3E,02,CD,8A     94F9
1520 DATA B1,AF,32,CA,B2,21,C9,B2     9DFF
1530 DATA 77,E5,CD,C6,B0,E1,34,20     A677
1540 DATA F8,C9,F5,C5,E5,CD,CB,B1     B9B4
1550 DATA C1,F1,5F,F1,F5,D5,E5,F5     CBA0
1560 DATA C5,D5,E5,78,CD,EC,B1,E1     DE45
1570 DATA D1,CD,E0,B1,0D,20,FA,C1     F32A
1580 DATA F1,3D,20,EB,E1,D1,F1,F5     64D
1590 DATA C5,D5,E5,3E,01,CD,EC,B1     CAB
1600 DATA CD,E0,B1,10,F6,E1,D1,CD     1288
1610 DATA D6,B1,0D,20,FA,C1,F1,3D     1A7E
1620 DATA 20,E5,C9,06,00,50,CD,11     2353
```

```
1630 DATA 01,CD,14,01,57,C9,CB,0A    303B
1640 DATA D0,C5,01,08,00,09,C1,C9    4264
1650 DATA 23,7D,E6,07,C0,C5,01,F8    534E
1660 DATA 00,09,C1,C9,C5,47,CD,4A    6941
1670 DATA 00,4F,7A,2F,A1,CB,03,30    709C
1680 DATA 01,B2,05,28,0C,CB,0A,30    888F
1690 DATA F0,CD,4D,00,CD,D9,B1,18    8ECA
1700 DATA E5,CD,4D,00,C1,C9,3A,CA    9560
1710 DATA B2,F5,E6,07,07,4F,07,81    A022
1720 DATA C6,BF,4F,F1,E6,38,0F,5F    A892
1730 DATA 0F,83,C6,07,5F,C9,3A,C9    B3EE
1740 DATA B2,F5,CD,3C,B2,C6,0C,4F    C0F7
1750 DATA F1,0F,0F,0F,0F,CD,3C,B2    E089
1760 DATA C6,08,5F,C9,E6,0F,57,07    F5D8
1770 DATA 47,07,07,80,82,C9,3A,C9    80D
1780 DATA B2,6F,26,00,29,29,29,EB    E64
1790 DATA FD,21,CB,B2,FD,19,C9,06    144C
1800 DATA 00,C5,D5,CD,78,B2,D1,C1    1BF4
1810 DATA 04,21,40,1F,CD,9C,00,20    2540
1820 DATA 07,2B,7C,B5,20,F6,18,E9    3298
1830 DATA CB,40,C4,78,B2,C3,9F,00    44A0
1840 DATA D5,CD,CB,B1,F1,47,5F,D5    5746
1850 DATA E5,CD,4A,00,AA,CD,4D,00    6C80
1860 DATA CD,D6,B1,1D,20,F3,E1,D1    80BD
1870 DATA CD,E0,B1,10,EA,C9,01,00    8827
1880 DATA 08,11,80,BB,ED,53,20,F9    906D
1890 DATA 21,CB,B2,ED,B0,CD,38,01    97C6
1900 DATA 07,07,E6,03,4F,06,00,21    A27B
1910 DATA C1,FC,09,CB,7E,28,0E,21    AA59
1920 DATA C5,FC,09,7E,07,07,07,07    BC87
1930 DATA E6,0C,B1,CB,FF,32,1F,F9    CE63
1940 DATA C9,00,00,01,24,79,1F,1F    E216
1950 DATA 1F,E6,07,E9,E5,2A,5E,EC    F6A1
1960 DATA 7E,23,22,5E,EC,E1,4F,C9    848
1970 DATA AF,BE,28,0D,79,BE,23,3E    EF9
1980 DATA 00,C8,CB,7E,23,28,FB,18    1385
1990 DATA EF,23,7E,23,A1,BE,23,7E    1D63
2000 DATA 23,28,0D,CB,7E,20,F3,7E    2672
2010 DATA 87,23,30,FB,CB,7E,20,EA    34AF
2020 DATA CB,7E,C8,23,23,23,18,F8    4641
2030 DATA E5,C5,21,33,63,01,05,00    5638
2040 DATA ED,B1,C1,E1,28,0D,C9,FE    6A63
2050 DATA E9,28,08,C9,FE,45,28,03    7F14
2060 DATA FE,4D,C0,F5,3E,FF,32,47    8974
2070 DATA EC,F1,C9,18,C3,76,C9,E9    8E8B
2080 DATA CD,D4,62,18,0B,CD,D4,62    9495
2090 DATA 47,CD,D4,62,CD,31,64,78    9E4F
2100 DATA CD,31,64,3E,48,C3,41,64    A6E8
2110 DATA 79,E6,38,18,F3,3E,49,CB    B8CA
```

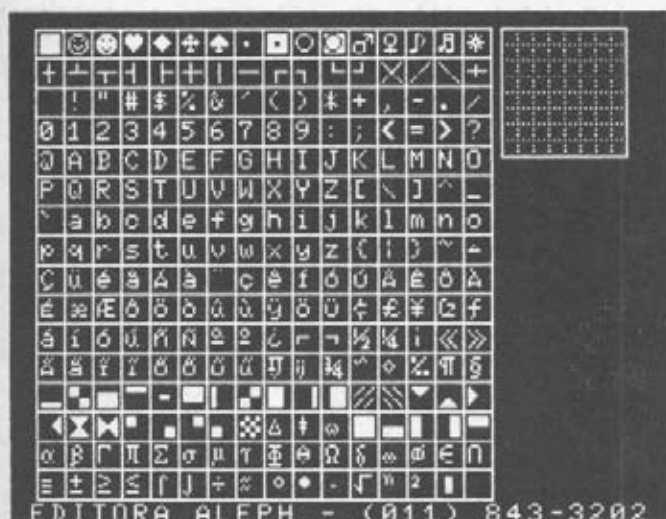
```
2120 DATA 59,28,02,3E,52,18,22,C3    C80F
2130 DATA 3A,64,3E,30,11,3E,31,11    D0F0
2140 DATA 3E,32,18,15,CD,29,64,CD    F3A6
2150 DATA 84,63,18,3F,CD,29,64,CD    512
2160 DATA 9B,63,18,37,CD,87,63,3E    B03
2170 DATA 27,C3,41,64,3E,0E,11,3E    F4E
2180 DATA 20,11,3E,26,11,3E,0A,11    1A0E
2190 DATA 3E,0C,11,3E,07,18,77,FE    2352
2200 DATA 06,28,EC,E6,3E,C6,08,FE    3206
2210 DATA 0C,20,6B,3A,55,EC,C6,0C    430F
2220 DATA 18,64,CD,29,64,CD,D4,62    5566
2230 DATA CD,48,63,3E,29,18,CA,CD    6AD9
2240 DATA 29,64,C5,CD,3D,63,C1,18    7ED9
2250 DATA F2,CD,29,64,CD,A3,63,18    870A
2260 DATA EA,E6,03,87,C6,10,18,3E    8D35
2270 DATA CD,D4,62,3A,5E,EC,81,47    94B6
2280 DATA 3A,5F,EC,CE,00,CB,79,28    9DB1
2290 DATA 01,3D,C3,44,63,79,E6,07    A68E
2300 DATA FE,06,20,22,CD,29,64,CD    B8CF
2310 DATA A3,63,3A,55,EC,B7,28,BB    CA60
2320 DATA C5,CD,D4,62,C1,B7,28,B3    DF89
2330 DATA 16,2B,F2,09,64,16,2D,ED    F562
2340 DATA 44,CD,2B,64,18,A2,21,AF    637
2350 DATA 64,85,6F,30,01,24,7E,E6    BDA
2360 DATA 7F,FE,20,C4,41,64,BE,F8    10B5
2370 DATA 23,18,F3,FD,21,D1,EC,18    1A60
2380 DATA ED,16,28,FD,72,00,FD,23    2280
2390 DATA C9,F5,0F,0F,0F,0F,CD,3A    3100
2400 DATA 64,F1,E6,0F,FE,0A,DE,69    43A3
2410 DATA 27,FD,77,00,FD,23,C9,3E    55E4
2420 DATA 0D,18,F6,FD,21,13,ED,3E    6BAD
2430 DATA 50,FD,2B,FD,36,00,20,3D    8003
2440 DATA 20,F7,C9,3B,3B,E5,3B,E8    88FC
2450 DATA 8A,8A,E8,9B,3B,5F,E5,E8    8EF6
2460 DATA 3B,90,9B,A3,9B,E8,E5,87    962A
2470 DATA 7C,50,3B,93,E5,62,3B,65    A08F
2480 DATA 3B,68,3B,84,A3,74,93,93    AA16
2490 DATA 74,93,55,55,93,C1,3B,97    BBAF
2500 DATA 3B,CB,3B,8D,90,6C,A3,93    CCC4
2510 DATA AA,AA,93,38,3B,93,38,E8    E36F
2520 DATA 38,9B,3D,C9,D0,D0,3B,CB    F7B0
2530 DATA 3D,3D,3B,93,B7,B7,93,A3    74E
2540 DATA B7,B7,A3,9B,B7,B7,9B,C2    E21
2550 DATA C3,C4,C5,C8,CC,CD,C1,42    14AB
2560 DATA C3,44,C5,48,CC,53,D0,4E    1DF3
2570 DATA DA,5A,A0,4E,C3,43,A0,50    25ED
2580 DATA CF,50,C5,50,A0,4D,A0,41    34FE
2590 DATA C6,49,D8,49,D9,28,43,A9    4631
2600 DATA 3F,43,43,C6,2F,43,50,CC    5720
```

```
2610 DATA 27,44,41,C1,F3,44,C9,FB      6DE9
2620 DATA 45,C9,D9,45,58,D8,76,48      82E1
2630 DATA 41,4C,D4,17,52,4C,C1,07      8B46
2640 DATA 52,4C,43,C1,1F,52,52,C1      8FCF
2650 DATA 0F,52,52,43,C1,37,53,43      9729
2660 DATA C6,00,FF,2A,26,FF,22,27      A003
2670 DATA FF,3A,24,FF,32,25,FF,F9      A8D8
2680 DATA 10,EF,02,11,EF,0A,12,C0      B8B3
2690 DATA 40,09,C7,06,1E,CF,01,1F      C8D8
2700 DATA 4C,C4,FF,CE,1D,F8,88,0C      DF9E
2710 DATA 41,44,C3,FF,C6,1D,F8,80      F4A1
2720 DATA 0C,CF,09,08,41,44,C4,FF      3F1
2730 DATA E6,1C,F8,A0,01,41,4E,C4      A35
2740 DATA FF,CD,23,C7,C4,22,43,41      F2F
2750 DATA 4C,CC,FF,FE,1C,F8,B8,01      1740
2760 DATA 43,D0,C7,05,06,CF,0B,04      2000
2770 DATA 44,45,C3,FF,10,21,44,4A      2FDC
2780 DATA 4E,DA,FF,EB,18,FF,E3,19      4215
2790 DATA FF,08,0A,45,D8,FF,DB,1A      5403
2800 DATA 49,CE,C7,04,06,CF,03,04      6981
2810 DATA 49,4E,C3,C7,C2,22,FF,C3      7DCE
2820 DATA 23,FF,E9,15,4A,D0,FF,18      861B
2830 DATA 21,E7,20,20,4A,D2,FF,D3      8AF1
2840 DATA 1B,4F,55,D4,FF,00,00,4E      926D
2850 DATA 4F,D0,FF,F6,1C,F8,B0,01      9B13
2860 DATA 4F,D2,CF,C1,16,50,4F,D0      A496
2870 DATA CF,C5,16,50,55,53,C8,FF      B6DD
2880 DATA C9,00,C7,C0,17,52,45,D4      C750
2890 DATA C7,C7,0B,52,53,D4,FF,DE      DC05
2900 DATA 1D,F8,98,0C,53,42,C3,FF      F0EC
2910 DATA D6,1C,F8,90,01,53,55,C2      C
2920 DATA FF,EE,1C,F8,A8,01,58,4F      627
2930 DATA D2,00,74,66,6F,53,20,79      967
2940 DATA 6E,69,54,00,C0,40,05,42      143C
2950 DATA 49,D4,C0,80,05,52,45,D3      1DBD
2960 DATA C0,C0,05,53,45,D4,F8,10      2E0B
2970 DATA 01,52,CC,F8,18,01,52,D2      3EA2
2980 DATA F8,00,01,52,4C,C3,F8,08      505F
2990 DATA 01,52,52,C3,F8,20,01,53      6768
3000 DATA 4C,C1,F8,28,01,53,52,C1      78C7
3010 DATA F8,38,01,53,52,CC,00,A9      828F
3020 DATA 43,50,C4,B9,43,50,44,D2      876B
3030 DATA A1,43,50,C9,B1,43,50,49      8F94
3040 DATA D2,AA,49,4E,C4,BA,49,4E      987A
3050 DATA 44,D2,A2,49,4E,C9,B2,49      A2DD
3060 DATA 4E,49,D2,A8,4C,44,C4,B8      B534
3070 DATA 4C,44,44,D2,A0,4C,44,C9      C594
3080 DATA B0,4C,44,49,D2,44,4E,45      DBBD
3090 DATA C7,BB,4F,54,44,D2,B3,4F      F0B1
```

```
3100 DATA 54,49,D2,AB,4F,55,54,C4    FF85
3110 DATA A3,4F,55,54,C9,4D,52,45    46A
3120 DATA 54,C9,45,52,45,54,CE,6F    7D2
3130 DATA 52,4C,C4,67,52,52,C4,00    129D
3140 DATA CF,4A,07,41,44,C3,FF,46    1AA6
3150 DATA 0D,FF,56,0E,FF,5E,0F,49    2AA5
3160 DATA CD,C7,40,02,49,CE,CF,4B    3D3B
3170 DATA 28,CF,43,29,F7,57,13,F7    5006
3180 DATA 47,14,4C,C4,C7,41,03,4F    66A7
3190 DATA 55,D4,CF,42,07,53,42,C3    7B3D
3200 DATA 00,2A,77,7E,21,96,22,9E    8225
3210 DATA 09,86,19,8E,23,A6,29,AE    871A
3220 DATA 34,B6,35,BE,36,28,39,E1    8E84
3230 DATA 46,E3,4E,E5,56,E9,5E,F9    9B2D
3240 DATA 66,6E,70,71,72,73,74,75    A384
3250 DATA 3F,BF,CD,F3,43,28,07,FE    B59C
3260 DATA 3F,28,14,C3,47,43,21,0B    C4F2
3270 DATA 67,CD,02,67,CD,9F,00,F5    D9AA
3280 DATA CD,B4,42,F1,FE,03,C8,21    E09F
3290 DATA E3,67,7E,B7,C8,CD,B6,42    F857
3300 DATA 23,18,F7,45,44,49,54,4F    84
3310 DATA 52,20,43,4F,4D,4D,41,4E    572
3320 DATA 44,0D,0A,4E,45,57,0D,4C    FD7
3330 DATA 49,53,54,20,5B,6E,31,5B    186F
3340 DATA 2D,6E,32,5D,5D,0D,4C,4C    2977
3350 DATA 49,53,54,20,5B,6E,31,5B    3A51
3360 DATA 2D,6E,32,5D,5D,0D,41,55    40BE
3370 DATA 54,4F,20,5B,6E,31,5B,2C    6436
3380 DATA 6E,32,5D,5D,0D,52,45,4E    7939
3390 DATA 55,4D,20,5B,6E,31,5B,2C    7F96
3400 DATA 6E,32,5B,2C,6E,33,5D,5D    846F
3410 DATA 5D,0D,44,45,4C,45,54,45    8D0E
3420 DATA 20,6E,31,5B,2D,6E,32,5D    96FD
3430 DATA 0D,46,49,4E,44,73,0D,53    A1F4
3440 DATA 45,41,52,43,48,73,0D,4C    B445
3450 DATA 53,45,41,52,43,48,73,0D    C3EE
3460 DATA 43,48,41,4E,47,45,64,73    DAD9
3470 DATA 31,64,5B,73,32,5D,0D,53    F008
3480 DATA 41,56,45,20,22,66,69,6C    FD46
3490 DATA 65,20,6E,61,6D,65,22,0D    297
3500 DATA 4C,4F,41,44,20,22,66,69    72C
3510 DATA 6C,65,20,6E,61,6D,65,22    1280
3520 DATA 0D,4D,45,52,47,45,20,22    187C
3530 DATA 66,69,6C,65,20,6E,61,6D    2000
3540 DATA 65,22,0D,4D,41,50,0D,41    3076
3550 DATA 5B,4E,55,50,4F,49,52,53    507E
3560 DATA 44,48,2F,78,78,5D,0D,42    66A7
3570 DATA 41,0D,00,0D,4D,4F,4E,49    7AFF
3580 DATA 54,4F,52,20,43,4F,4D,4D    8184
```

```
3590 DATA 41,4E,44,0D,0A,43,78,20
3600 DATA 20,20,20,20,20,20,20,20
3610 DATA 20,20,20,43,68,61,6E,67
3620 DATA 65,20,64,75,6D,70,0D,44
3630 DATA 78,78,5B,2C,79,79,5D,20
3640 DATA 20,20,20,20,20,44,75,6D
3650 DATA 70,20,6D,65,6D,6F,72,79
3660 DATA 0D,50,78,78,5B,2C,79,79
3670 DATA 5D,20,20,20,20,20,20,50
3680 DATA 72,69,6E,74,20,6D,65,6D
3690 DATA 6F,72,79,0D,56,78,78,5B
3700 DATA 2C,79,79,5D,20,20,20,20
3710 DATA 20,20,56,2D,52,41,4D,20
3720 DATA 70,72,69,6E,74,0D,4D,5B
3730 DATA 78,78,5D,20,20,20,20,20
3740 DATA 20,20,20,20,4D,65,6D,6F
3750 DATA 72,79,20,73,65,74,0D,53
3760 DATA 5B,78,78,5D,20,20,20,20
3770 DATA 20,20,20,20,20,53,65,74
3780 DATA 20,6D,65,6D,6F,72,79,0D
3790 DATA 46,78,78,2C,79,79,2C,7A
3800 DATA 20,20,20,20,20,20,46,69
3810 DATA 6C,6C,20,6D,65,6D,6F,72
3820 DATA 79,0D,54,78,78,2C,79,79
3830 DATA 2C,7A,7A,20,20,20,20,20
3840 DATA 54,72,61,6E,73,66,65,72
3850 DATA 0D,52,5B,78,78,5D,20,20
3860 DATA 20,20,20,20,20,20,20,52
3870 DATA 65,61,64,20,74,61,70,65
3880 DATA 0D,58,5B,72,5D,20,20,20
3890 DATA 20,20,20,20,20,20,20,65
3900 DATA 78,61,6D,69,6E,65,20,72
3910 DATA 65,67,2E,0D,47,78,78,5B
3920 DATA 2C,79,79,5B,2C,7A,7A,5D
3930 DATA 5D,20,47,6F,0D,4C,5B,50
3940 DATA 5D,5B,78,78,5B,2C,79,79
3950 DATA 5D,5D,20,64,69,73,61,73
3960 DATA 73,65,6D,62,6C,65,0D,3F
3970 DATA 5B,3F,5D,20,20,20,20,20
3980 DATA 20,20,20,20,20,64,69,73
3990 DATA 70,6C,61,79,20,74,68,69
4000 DATA 73,0D,0D,00,DB,A8,32,32
4010 DATA FA,3A,C1,FC,32,30,FA,32
4020 DATA 31,FA,0E,00,CD,95,69,38
4030 DATA 03,32,30,FA,0E,40,CD,95
4040 DATA 69,38,03,32,31,FA,21,C9
4050 DATA FC,06,40,7E,87,38,2B,23
4060 DATA 10,F9,2A,48,FC,11,00,80
4070 DATA B7,ED,52,20,1D,21,30,FA
```

```
4080 DATA 3A,C1,FC,BE,28,14,23,BE
4090 DATA 28,10,21,F0,69,11,00,C1
4100 DATA 01,C4,00,ED,B0,CD,00,C1
4110 DATA B7,C9,FB,37,C9,21,C1,FC
4120 DATA 06,04,AF,E6,03,B6,C5,E5
4130 DATA 61,2E,10,F5,CD,0C,00,2F
4140 DATA 5F,F1,D5,F5,CD,14,00,F1
4150 DATA D1,F5,D5,CD,0C,00,C1,47
4160 DATA 79,2F,5F,F1,F5,C5,CD,14
4170 DATA 00,C1,79,B8,20,17,F1,2D
4180 DATA 20,D9,24,24,24,24,4F,7C
4190 DATA FE,40,28,05,FE,80,79,20
4200 DATA C8,79,E1,E1,C9,F1,E1,C1
4210 DATA B7,F2,EA,69,C6,04,FE,90
4220 DATA 38,B4,23,3C,10,AD,37,C9
4230 DATA 3A,30,FA,26,00,CD,1C,C1
4240 DATA 3A,31,FA,26,40,CD,1C,C1
4250 DATA DB,A8,F5,3A,32,FA,D3,A8
4260 DATA 47,F1,FB,C9,CD,49,C1,FA
4270 DATA 29,C1,DB,A8,A1,B0,D3,A8
4280 DATA C9,CD,98,C1,28,13,E5,CD
4290 DATA 6E,C1,4F,06,00,7D,A4,B2
4300 DATA 21,C5,FC,09,77,E1,79,18
4310 DATA DB,CD,A1,C1,21,C5,FC,72
4320 DATA C9,F3,F5,7C,07,07,E6,03
4330 DATA 5F,1C,3E,C0,07,07,1D,20
4340 DATA FB,5F,2F,4F,F1,F5,E6,03
4350 DATA 47,04,3E,A8,C6,55,10,FC
4360 DATA 57,A3,47,F1,B7,C9,F5,7A
4370 DATA E6,C0,4F,F1,F5,57,DB,A8
4380 DATA 47,E6,3F,B1,F5,7A,0F,0F
4390 DATA E6,03,57,14,3E,AB,C6,55
4400 DATA 15,20,FB,A3,57,7B,2F,67
4410 DATA F1,CD,B4,C1,F1,E6,03,C9
4420 DATA 14,15,C0,47,7B,FE,03,78
4430 DATA C9,0F,0F,E6,03,57,3A,FF
4440 DATA FF,2F,47,E6,FC,B2,57,32
4450 DATA FF,FF,7B,C9,D3,A8,3A,FF
4460 DATA FF,2F,6F,A4,B2,32,FF,FF
4470 DATA 78,D3,A8,C9,0D,2A,2A,20
4480 DATA 43,4F,4D,50,49,4C,41,44
4490 DATA 4F,52,20,41,53,4D,43,4F
4500 DATA 43,41,52,BA,BB,B6,AB,B0
4510 DATA AD,BE,DF,BE,B3,BA,AF,B7
4520 DATA DF,D2,DF,D7,CF,CE,CE,D6
4530 DATA DF,C7,CB,CC,D2,CC,CD,CF
4540 DATA CD,FF,20,00,00,00,00,00
```

Com o programa digitado e gravado corretamente, comande RUN para executá-lo. A tela deverá estar como mostra a figura a seguir:

Você tem agora em seu micro um poderoso editor de caracteres com dois modos de operação: SELEÇÃO e EDIÇÃO. O modo SELEÇÃO permite a escolha do caractere a ser editado. O modo EDIÇÃO permite a alteração do seu "desenho". Logo após ser carregado, o programa opera no modo SELEÇÃO. Experimente usar as teclas de setas e observe o que acontece com o cursor (na tabela de caracteres) e com o quadrado no canto superior direito do vídeo (CARACTERE AMPLIADO).

Para alterar ou redesenhar completamente um dado caractere, deve-se inicialmente levar o cursor até ele com a ajuda das teclas de setas.

Feito isso, deve-se entrar no modo de EDIÇÃO, pressionando a tecla RETURN. Assim procedendo, o cursor desaparecerá da tela e um pequeno ponto será visível no quadrado do CARACTERE AMPLIADO, onde poderá ser feita a edição.

Para apagar os pontos marcados do caractere a ser editado, basta pressionar a BARRA DE ESPAÇOS.

Para marcar um ponto no caractere, basta pressionar a tecla da letra "M" (de Marcar).

Após redesenhar o caractere, para voltar ao modo de SELEÇÃO, basta pressionar RETURN novamente.

Uma vez alterados ou redesenhados os caracteres, deve-se avisar ao programa que essa nova tabela deve ser usada. Para isso basta pressionar a tecla da letra "C" (de Confirmar).

Por fim, para sair do programa e retornar ao BASIC, pressiona-se a tecla da letra "S" (de Sair).

Resumindo, temos os seguintes comandos à nossa disposição:

SETAS - Movem o cursor;
C - Confirma o uso da tabela redefinida;
S - Sai do programa e retorna ao BASIC;
RETURN - Passa do modo SELEÇÃO para o EDIÇÃO e vice-versa;
ESPAÇO - Apaga pontos no caractere em edição;
M - Marca pontos no caractere em edição.

Até agora vimos como usar o programa para gerar e assumir uma nova tabela de caracteres. Vamos aprender como usar essa nova tabela.

Após ter retornado ao BASIC é conveniente, antes de mais nada, salvar a nova tabela em fita ou em disco. Para isso, comande:

```
BSAVE "TABELA.DAT",&HBB80,&HC380
```

A seguir, apague o programa em BASIC que está na memória do micro comandando NEW e SCREEN 1.

Caso você queira carregar uma tabela já salva em fita ou em disco e assumí-la, basta comandar:

```
BLOAD"TABELA.DAT"
POKE &HF920,&H80
POKE &HF921,&HBB
```

Só para o EXPERT:

```
POKE &HF91F,2
```

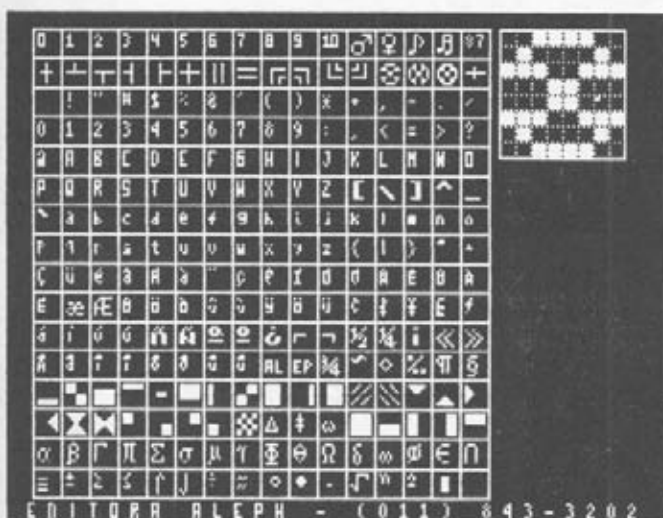
Só para o HOTBIT:

```
POKE &HF91F,3
```

A seguir, deve-se usar o comando SCREEN.

Para voltar a operar com a tabela original da ROM, rode o programinha mostrado a seguir.

```
10 POKE &HF91F,0
20 POKE &HF920,&HBF
30 POKE &HF921,&H1B
40 SCREEN 1
```

Agora experimente redefinir a tabela de caracteres como mostrado abaixo:

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.1 - páginas 83 a 86.
Coleção de Programas para MSX v.2 - páginas 10 a 24, 73 e 83 a 85.
Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - página 145.

2.C - SCROLL UP PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para cima. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

```
100 REM -------------------------------      65E
110 REM SCROLL SCREEN 0 UP - Rubens Jr.     10C6
120 REM -------------------------------     1EC6
130 FOR F=&HE000 TO &HE046                  2613
140    READ A$ : POKE F,VAL("&H"+A$)        3554
150 NEXT F : DEFUSR0=&HE000                 3C5D
160 DATA 21,00,00,22,44,E0,06,17            4AFE
170 DATA F3,21,28,00,ED,5B,44,E0            606D
180 DATA 19,C5,CD,2F,E0,C1,2A,44            74E1
190 DATA E0,11,28,00,19,22,44,E0            8161
200 DATA 10,E7,21,98,03,01,28,00            8629
210 DATA 3E,20,CD,56,00,FB,C9,E5            8CD8
220 DATA D5,01,28,00,C5,11,18,FC            9784
230 DATA D5,CD,59,00,E1,C1,D1,CD            9FE0
240 DATA 5C,00,E1,C9,00,00,00,00            B131
250 REM -------------------------------     C541
260 REM          Exemplo de uso             D752
270 REM -------------------------------     F4A8
280 SCREEN 0 : WIDTH 40 : KEY OFF           FF9E
290 X = 40 * RND(1)                          4CC
300 LOCATE X,20 : PRINT ".";                 7E8
310 X = USR0(0)                              802
320 GOTO 290                                 D07

                              TOTAL  =       D07
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.D - SCROLL DOWN PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para baixo. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

```
100 REM -------------------------------- 63E
110 REM SCROLL SCREEN 0 DOWN-Rubens Jr.  1189
120 REM -------------------------------- 1F89
130 FOR F=&HE000 TO &HE049               26E3
140    READ A$ : POKE F,VAL("&H"+A$)     3624
150 NEXT F : DEFUSR0=&HE000              3D2D
160 DATA 21,70,03,22,47,E0,06,17         4BD1
170 DATA F3,21,28,00,ED,5B,47,E0         6143
180 DATA 19,EB,C5,CD,32,E0,C1,2A         75A3
190 DATA 47,E0,11,28,00,B7,ED,52         8254
200 DATA 22,47,E0,10,E4,21,00,00         8750
210 DATA 01,28,00,3E,20,CD,56,00         8D4B
220 DATA FB,C9,E5,D5,01,28,00,C5         96F4
230 DATA 11,18,FC,D5,CD,59,00,E1         9F9D
240 DATA C1,D1,CD,5C,00,E1,C9,00         B15B
250 DATA 00,00,00,00,00,00,00,00         C11B
260 REM -------------------------------- DBC1
270 REM          Exemplo de uso          EE0A
280 REM -------------------------------- 101
290 SCREEN 0 : WIDTH 40 : KEY OFF        661
300 X = 40 * RND(1)                      A81
310 LOCATE X,0 : PRINT ".";              EA1
320 X = USR0(0)                          1284
330 GOTO 300                             153F

                          TOTAL =        153F
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.E - SCROLL LEFT PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para a esquerda. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

```
100 REM ---------------------------------- 65E
110 REM SCROLL SCREEN 0 LEFT-Rubens Jr.    1190
120 REM ---------------------------------- 1F90
130 FOR F=&HE000 TO &HE03F                 26F4
140   READ A$ : POKE F,VAL("&H"+A$)        3635
150 NEXT F : DEFUSR0=&HE000                303E
160 DATA 21,00,00,22,3D,E0,06,18           4C43
170 DATA F3,C5,CD,1C,E0,C1,2A,3D           6014
180 DATA E0,11,28,00,19,22,3D,E0           7560
190 DATA 10,EF,FB,C9,E5,01,28,00           81C7
200 DATA C5,11,18,FC,D5,CD,59,00           8700
210 DATA 21,19,FC,11,18,FC,01,27           8E3A
220 DATA 00,ED,B0,3E,20,12,E1,C1           981D
230 DATA D1,CD,5C,00,C9,00,00,E1           A08F
240 REM ---------------------------------- B764
250 REM          Exemplo de uso            C7F8
260 REM ---------------------------------- E54D
270 SCREEN 0 : WIDTH 40 : KEY OFF          F191
280 X = 23 * RND(1)                        FC7A
290 LOCATE 39,X : PRINT ".";               20D
300 X = USR0(0)                            549
310 GOTO 280                               693

                         TOTAL  =          693
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.F - SCROLL RIGHT PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para a direita. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

```
100 REM --------------------------------  65E
110 REM SCROLL SCREEN 0 RIGHT-Rubens Jr  1139
120 REM --------------------------------  1F39
130 FOR F=&HE000 TO &HE03F                269D
140   READ A$ : POKE F,VAL("&H"+A$)       350E
150 NEXT F : DEFUSR0=&HE000               3CE7
160 DATA 21,00,00,22,3D,E0,06,18          4BEC
170 DATA F3,C5,CD,1C,E0,C1,2A,3D          5F8D
180 DATA E0,11,28,00,19,22,3D,E0          7509
190 DATA 10,EF,FB,C9,E5,01,28,00          8170
200 DATA C5,11,18,FC,D5,CD,59,00          8779
210 DATA 21,3E,FC,11,3F,FC,01,27          8ECD
220 DATA 00,ED,B8,3E,20,12,E1,C1          98A8
230 DATA D1,CD,5C,00,C9,00,00,E1          A11A
240 REM --------------------------------  B7EF
250 REM          Exemplo de uso           C883
260 REM --------------------------------  E508
270 SCREEN 0 : WIDTH 40 : KEY OFF         F21C
280 X = 23 * RND(1)                       FD05
290 LOCATE 0,X : PRINT ".";               269
300 X = USR0(0)                           59A
310 GOTO 280                              6D3

                              TOTAL =     6D3
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.G - SCROLL UP PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para cima. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

```
100 REM -----------------------------------  65E
110 REM SCROLL SCREEN 1 UP - Rubens Jr.      10C7
120 REM -----------------------------------  1EC7
130 FOR F=&HE000 TO &HE046                   2614
140   READ A$ : POKE F,VAL("&H"+A$)          3555
150 NEXT F : DEFUSR0=&HE000                  3C5E
160 DATA 21,00,18,22,44,E0,06,17             4AF8
170 DATA F3,21,20,00,ED,5B,44,E0             606F
180 DATA 19,C5,CD,2F,E0,C1,2A,44             74E3
190 DATA E0,11,20,00,19,22,44,E0             8168
200 DATA 10,E7,21,E0,1A,01,20,00             86A2
210 DATA 3E,20,CD,56,00,FB,C9,E5             8D54
220 DATA D5,01,20,00,C5,11,18,FC             9805
230 DATA D5,CD,59,00,E1,C1,D1,CD             A061
240 DATA 5C,00,E1,C9,00,00,C9,00             B1EC
250 REM -----------------------------------  C5FC
260 REM          Exemplo de uso              D800
270 REM -----------------------------------  F563
280 SCREEN 1 : WIDTH 32 : KEY OFF            54
290 X = 32 * RND(1)                          58A
300 LOCATE X,20 : PRINT ".";                 8A9
310 X = USR0(0)                              BC0
320 GOTO 290                                 DC5

                              TOTAL =        DC5
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

2.H - SCROLL DOWN PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para baixo. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

```
100 REM -------------------------------- 65E
110 REM SCROLL SCREEN 1 DOWN-Rubens Jr.  118A
120 REM -------------------------------- 1F8A
130 FOR F=&HE000 TO &HE049               26E4
140   READ A$ : POKE F,VAL("&H"+A$)      3B25
150 NEXT F : DEFUSR0=&HE000              3D2E
160 DATA 21,C0,1A,22,47,E0,06,17         4C6D
170 DATA F3,21,20,00,ED,5B,47,E0         61E7
180 DATA 19,EB,C5,CD,32,E0,C1,2A         7647
190 DATA 47,E0,11,20,00,B7,ED,52         82F0
200 DATA 22,47,E0,10,E4,21,00,18         87E3
210 DATA 01,20,00,3E,20,CD,56,00         8D06
220 DATA FB,C9,E5,D5,01,20,00,C5         9787
230 DATA 11,18,FC,D5,CD,59,00,E1         A023
240 DATA C1,D1,CD,5C,00,E1,C9,00         B1EE
250 DATA 00,0E,00,00,00,00,00,00         C1D9
260 REM -------------------------------- DC7F
270 REM          Exemplo de uso          EF98
280 REM -------------------------------- 13F
290 SCREEN 1 : WIDTH 32 : KEY OFF        728
300 X = 40 * RND(1)                      B78
310 LOCATE X,0 : PRINT ".";              F68
320 X = USR0(0)                          134B
330 GOTO 300                             1606

                              TOTAL  =   1606
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

2.1 - SCROLL LEFT PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para a esquerda. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

```
100 REM --------------------------------       65E
110 REM SCROLL SCREEN 1 LEFT-Rubens Jr.        1191
120 REM --------------------------------       1F91
130 FOR F=&HE000 TO &HE03F                     26F5
140   READ A$ : POKE F,VAL("&H"+A$)            3636
150 NEXT F : DEFUSR0=&HE000                    3D3F
160 DATA 21,00,18,22,3D,E0,06,18               4C3D
170 DATA F3,C5,CD,1C,E0,C1,2A,3D               600E
180 DATA E0,11,20,00,19,22,3D,E0               7552
190 DATA 10,EF,FB,C9,E5,01,20,00               81C1
200 DATA C5,11,18,FC,D5,CD,59,00               87CA
210 DATA 21,19,FC,11,18,FC,01,1F               8DC8
220 DATA 00,ED,B0,3E,20,12,E1,C1               97AB
230 DATA D1,CD,5C,00,C9,00,00,E1               A01D
240 REM --------------------------------       B6F2
250 REM          Exemplo de uso                C786
260 REM --------------------------------       E4DB
270 SCREEN 1 : WIDTH 32 : KEY OFF              F114
280 X = 23 * RND(1)                            FBFD
290 LOCATE 31,X : PRINT ".";                   188
300 X = USR0(0)                                4C4
310 GOTO 280                                   60E

                                   TOTAL =     60E
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

2.J - SCROLL RIGHT PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para a direita. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

```
100 REM ---------------------------------- 63E
110 REM SCROLL SCREEN 1 RIGHT-Rubens Jr   113A
120 REM ---------------------------------- 1F3A
130 FOR F=&HE000 TO &HE03F                 269E
140   READ A$ : POKE F,VAL("&H"+A$)        350F
150 NEXT F : DEFUSR0=&HE000                3CE8
160 DATA 21,00,18,22,3D,E0,06,18           4BE6
170 DATA F3,C5,CD,1C,E0,C1,2A,3D           5F87
180 DATA E0,11,20,00,19,22,3D,E0           74F8
190 DATA 10,EF,FB,C9,E5,01,20,00           816A
200 DATA C5,11,18,FC,D5,CD,59,00           8773
210 DATA 21,36,FC,11,37,FC,01,1F           807F
220 DATA 00,ED,B8,3E,20,12,E1,C1           975A
230 DATA D1,CD,5C,00,C9,00,00,E1           9FCC
240 REM ---------------------------------- B6A1
250 REM           Exemplo de uso           C735
260 REM ---------------------------------- E48A
270 SCREEN 1 : WIDTH 32 : KEY OFF          F0C3
280 X = 23 * RND(1)                        F8AC
290 LOCATE 0,X : PRINT ".";                110
300 X = USR0(0)                            441
310 GOTO 280                               57A

                           TOTAL  =        57A
```

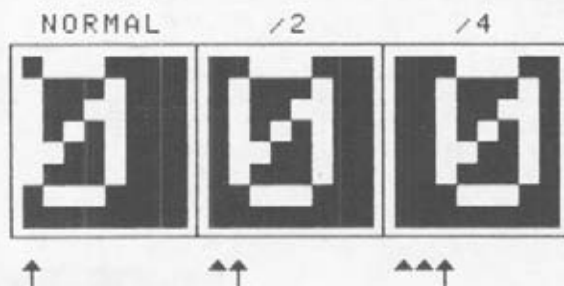
BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

2.K - CENTRALIZANDO CARACTERES

Os caracteres são desenhados dentro de uma matriz de 8x8 posições. Muitos deles usam apenas o lado esquerdo dessa matriz e há casos em que ao serem impressos na SCREEN 1 ou SCREEN 2 ficam fora de alinhamento. Podemos evitar isso de uma forma bem simples e rápida, redefinindo os caracteres. A título de exemplo, vamos 'centralizar' os caracteres dos números e das letras maiúsculas na SCREEN 1.

```
100 SCREEN 1                                     168
110 FOR F=6144 TO 6144+24*32-1 STEP 2            9F2
120   VPOKE F,ASC("█")                           E70
130 NEXT F : LOCATE 10,10                        1296
140 PRINT "0123456789" : LOCATE 2,12             1DEB
150 PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"           2553
160 FOR F=ASC("0") TO 8*ASC("9")+7               351C
170   VPOKE F,VPEEK(F)/2                         3DA3
180 NEXT F                                       3FE8
190 FOR F=ASC("A") TO 8*ASC("Z")+7               41A6
200   VPOKE F,VPEEK(F)/2                         5672
210 NEXT F                                       591E

                             TOTAL      =        591E
```

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.2 - páginas 72 e 73.
Aprofundando-se no MSX - página 90.

2.L - ANIMAÇÃO COM SPRITES

Se você definir vários SPRITE's com uma figura em posições sucessivas e os colocar na tela sempre na mesma camada, em sequência, elas darão o efeito de animação.

Se você quiser alterar o desenho do "boneco" no programa listado a seguir, uma boa idéia é eliminar a linha 90, digitando seu conteúdo manualmente no modo direto. A seguir você interrompe a listagem na sequência de linhas DATA que definem o SPRITE a ser alterado, modifica o desenho e comanda RUN. Desta forma você pode congelar o boneco com STOP e alterar os 0's e 1's da linha DATA, visualizando o efeito anterior à alteração.

Obviamente você pode definir SPRITE's de 16x16 pixels, obtendo figuras muito mais detalhadas.

Esse programa é apenas um exemplo (parece um bêbado tentando dançar BREAK!) mas pode ser alterado e melhorado conforme sua imaginação.

Pegue papel quadriculado e lápis e monte sua história!

```
90 SCREEN 1
100 GOSUB 220:SPRITE$(0)=S$
110 GOSUB 220:SPRITE$(1)=S$
120 GOSUB 220:SPRITE$(2)=S$
130 GOSUB 220:SPRITE$(3)=S$
140 GOSUB 220:SPRITE$(4)=S$
150 FOR I=1 TO 250 STEP 10
160 FOR S=0 TO 4
170 PUT SPRITE 0,(I+S*2,30),8,S
180 FORT=0TO50:NEXTT
190 NEXT S
200 NEXT I
210 GOTO 150
220 S$=""
230 FOR C=1 TO 8
240 READ K$
250 S$=S$+CHR$(VAL("&B"+K$))
260 NEXT C
270 RETURN
280 DATA 00011000
290 DATA 00011000
```

```
300 DATA 00111100                 831E
310 DATA 00111100                 8589
320 DATA 00111100                 87E6
330 DATA 00011000                 8A25
340 DATA 00011000                 8C3E
350 DATA 00011100                 8DE0
360 REM----------                 92F4
370 DATA 00011000                 9819
380 DATA 00011000                 9D28
390 DATA 00111100                 A261
400 DATA 01011010                 A72C
410 DATA 01011010                 AE49
420 DATA 00010100                 B562
430 DATA 00100110                 BC42
440 DATA 00110000                 C245
450 REM----------                 C92C
460 DATA 00011000                 D05D
470 DATA 00011000                 DA24
480 DATA 00111100                 E391
490 DATA 01011010                 EC8C
500 DATA 10011001                 F595
510 DATA 00100100                 FE24
520 DATA 01000010                 268
530 DATA 01100011                 48A
540 REM----------                 6CF
550 DATA 00011000                 875
560 DATA 00011000                 9C1
570 DATA 00111100                 E08
580 DATA 01011010                 1323
590 DATA 10011010                 182F
600 DATA 01100100                 1C74
610 DATA 01000100                 2058
620 DATA 00000110                 25D8
630 REM----------                 2ACE
640 DATA 00011000                 3476
650 DATA 00011000                 3AA0
660 DATA 00111100                 408C
670 DATA 00111100                 46C2
680 DATA 01011010                 5002
690 DATA 00101000                 5A76
700 DATA 00101100                 640B
710 DATA 00111110                 6D1A
720 REM----------                 7715
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.M - LETRAS AMPLIADAS

O programa apresentado a seguir gera uma tela padrão na SCREEN 2 onde podem ser inseridas 4 mensagens. Digite-o e execute-o. Depois, experimente alterar o conteúdo das linhas 135, 140, 145 e 150. Experimente também alterar o caractere entre as aspas na linha 130. A tela gerada será gravada em disco ou em fita pela linha 705. Se desejar, altere o nome do arquivo.

```
100 COLOR 15,1,1                              10E
105 SCREEN 2                                  39E
110 OPEN"GRP:" AS #1                          769
115 REM                                       8A8
120 REM define mensagens                      E2A
125 REM ---------------------------------     1A43
130 X$(0)="O"                                 1E8E
135 X$(1)="XXXXX"                             2370
140 X$(2)="XXXXXXXXXXXX"                      2EC7
145 X$(3)="XXXXXXXXXXXX"                      30FD
150 X$(4)="XXXXXXXXXXXXXXXXXXXXXXXXXXXX"      5C7B
155 REM                                       5DE8
160 REM desenha mensagem                      6B8A
165 REM ---------------------------------     8601
170 X=6.5+(245-8*LEN(X$(4)))/2                9204
175 PSET(X,170),POINT(X,170)                  97E8
180 PRINT #1, X$(4)                           9AC9
185 PSET(X+1,170),POINT(X+1,170)              A42C
190 PRINT #1, X$(4)                           AADE
195 REM                                       AB06
200 REM desenha linhas horizontais            AFD5
205 REM ---------------------------------     C626
210 COLOR 9                                   C8AD
215 YY = 1                                    CDC8
220 FOR F=6 TO 191 STEP .2                    D919
225   F = F + YY                              E105
230   LINE(8,F)-(247,F),9                     EF8F
235   YY = YY + .4                            FCA6
240 NEXT F                                    FF90
245 REM                                       101
250 REM desenha a moldura                     A68
255 REM ---------------------------------     FAF
260 FOR F=0 TO 255 STEP 8                     1594
265   PSET(F,0),4                             198D
270   PRINT #1,X$(0)                          1F38
275   PSET(F,191-7),4                         2500
280   PRINT #1,X$(0)                          2B38
285 NEXT F                                    2DAC
```

```
290 FOR F=0 TO 191 STEP 8
295   PSET(0,F),4
300   PRINT #1,X$(0)
305   PSET(255-7,F),4
310   PRINT #1,X$(0)
315 NEXT F
320 REM
325 REM desenha texto
330 REM ---------------------------------
335 COLOR 14
340 EN=PEEK(&HF920)+256*PEEK(&HF921)
345 FOR TX=1 TO 3
350   A$=X$(TX)
355   Q=0
360   FOR F=1 TO LEN(A$)
365    A=EN+8*ASC(MID$(A$,F,1))
370    FOR G=0 TO 7
375     B$=BIN$(PEEK(A+G))
380     B$=RIGHT$("00000000"+B$,8)
385     FOR H=1 TO 8
390       IF MID$(B$,H,1)="1" THEN Q=Q+1
395     NEXT H
400    NEXT G
405   NEXT F
410   P=2*Q-1
415   DIM X(P),Y(P),S(P),T(P)
420   E=0
425   I=5
430   FOR F=1 TO LEN(A$)
435    A=EN+8*ASC(MID$(A$,F,1))
440    FOR G=0 TO 7
445     B$=BIN$(PEEK(A+G))
450     B$=RIGHT$("00000000"+B$,8)
455     FOR H=1 TO 8
460      IF MID$(B$,H,1)="0" THEN 490
465      X(E)=I+H-1
470      Y(E)=8-G
475      X(E+Q)=X(E)
480      Y(E+Q)=Y(E)
485      E=E+1
490     NEXT H
495    NEXT G
500    REM
505    REM passo horizontal
510    REM ----------------------------
515    PS=6
520    I=I + PS
525   NEXT F
530   REM
```

```
535    REM parâmetro de escala X
540    REM ---------------------------
545    PX=3
550    IF TX=1 THEN PX=5
555    REM
560    REM parâmetro de escala Y
565    REM ---------------------------
570    PY=5
575    IF TX=1 THEN PY=7
580    REM
585    REM acha posições na tela
590    REM ---------------------------
595    XI=INT((230-LEN(A$)*PS*PX)/2)
600    IF TX=1 THEN XI=XI-8
605    YI=30+45*TX
610    IF TX=1 THEN YI=YI-1
615    IF TX=2 THEN YI=YI-1
620    IF TX=3 THEN YI=YI+5
625    DEFFN A(F)=XI+PX*X(F)
630    DEFFN B(F)=YI-PY*Y(F)
635    REM
640    REM "plota" círculos das letras
645    REM ---------------------------
650    FOR F=0 TO P-1
655     CX=FN A(F) :REM x do centro
660     CY=FN B(F) :REM y do centro
665     AX=1.7     :REM achatamento
670     RD=4       :REM raio
675     IF TX=1 THEN AX=1.4 : RD=6
680     CIRCLE(CX,CY),RD,,,,AX
685     PAINT(CX,CY)
690    NEXT F
695    ERASE X,Y,S,T
700 NEXT TX
705 BSAVE"TELAXXXX.SCR",0,&H3FFF,S
710 GOTO 710

                        TOTAL  =    2A3A
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.N - ENTENDENDO O COMANDO DRAW

O comando DRAW do BASIC MSX é, na realidade, uma macro-linguagem gráfica que permite a confecção de desenhos nas telas 2 e 3.

Sua sintaxe é simples:

DRAW <expressão string>

A montagem do desenho é feita através da expressão string segundo regras bem determinadas e de fácil memorização. Para conhecê-los basta consultar o verbete DRAW do dicionário de comandos do livro LINGUAGEM BASIC MSX.

A melhor maneira de se familiarizar com esse comando é gerar uma variável string contendo os comandos desejados e ver o efeito na tela.

Para facilitar a visualização do desenho obtido é conveniente "reticular" a tela (de 10 em 10 pontos, por exemplo) antes da execução do desenho. Experimente digitar o programa a seguir:

```
10 SCREEN 2                                 19D
20 FOR L=0 TO 191 STEP 10                   6B8
30   FOR C=0 TO 255 STEP 10                 CA2
40     PSET (C,L)                           F42
50   NEXT C                                 1104
60 NEXT L                                   1277
70 A$="BM60,60U10F10D10L5U5L5D5L10"         1C21
80 DRAW A$                                  1E53
90 GOTO 90                                  231E

                          TOTAL =           231E
```

Agora vá alterando a variável A$ definida na linha 80 com novos sub-comandos do DRAW e veja os efeitos gerados por suas experiências até se familiarizar com esse poderoso comando.

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 56 a 58.
Coleção de Programas para MSX v.1 - página 54.
Coleção de Programas para MSX v.2 - página 52.
Curso de BASIC v.1 - páginas 61 e 62.

2.0 - "WARP 8" NA SCREEN 2

Você está indo para o planeta da Princesa Vespa para salvá-la do terrível "Capacete Preto"! Para chegar lá você passa por um aglomerado de estrelas, com velocidade hiperfotônica. Como simular este efeito? Digite o programa a seguir e boa viagem!

```
100 SCREEN 2:DEFINT I-S,X-Y                              5C0
110 SPRITE$(1)=CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+C         1E30
HR$(0)+CHR$(8)+CHR$(0)+CHR$(0)+CHR$(0)
120 DIM X(31),Y(31),C(31),D(31),VX(31),V                 3094
Y(31),X1(31),Y1(31)
130 FOR I=0 TO 31                                        3746
140 X(I)=80+INT(80*RND(-TIME))                           4380
150 FOR T=0 TO INT(300*RND(3)):NEXTT                     4FF6
160 Y(I)=40+INT(80*RND(-TIME))                           5C04
170 IF X(I)=125 THEN GOTO 140                            6994
180 X(0)=126:Y(0)=86                                     7556
190 PUT SPRITE I,(X(I),Y(I)),15,1                        7E33
200 D(I)=(Y(I)-85)/(X(I)-125)                            876A
210 NEXT I                                               89A6
220 FOR T=1 TO 1000                                      8F30
230 FOR I=0 TO 31                                        94BE
240 VX(I)=SGN(X(I)-125)*I*T^3/1000                       A06E
250 VY(I)=SGN(Y(I)-85)*ABS(VX(I)*D(I))                   AF00
260 X1(I)=X(I)+VX(I):Y1(I)=Y(I)+VY(I)                    C748
270 ON ERROR GOTO 330                                    CD19
280 IF X1(I)<0 OR X1(I)>225 OR Y1(I)<0 O                 ED32
R Y1(I)>191 THEN C=14
290 PUT SPRITE I,(X1(I),Y1(I)),15-C,1                    115
300 C=0                                                  284
310 NEXT I                                               381
320 NEXT T                                               520
330 GOTO 330                                             6E8

                                      TOTAL   =          6E8
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.P - "SPRITEANDO" A TABELA DE CARACTERES

Muitas vezes sentimos a necessidade de fazer letras ou símbolos se movimentarem na tela como se fossem SPRITES. Isso é simples de ser feito: como podemos definir 256 SPRITES (8x8) e o MSX dispõe de 256 caracteres, podemos transformar cada caractere num SPRITE!

O programa a seguir procura o endereço do começo da tabela de caracteres na váriavel EN (contida nos endereços &HF920 e &HF921) e transfere sua configuração para a região da VRAM reservada para os desenhos dos SPRITES.

```
10 SCREEN 1 : DEFINT A-Z : KEY OFF                6A1
20 EN!=PEEK(&HF920)+256*PEEK(&HF921)             1030
30 FOR F=0 TO 2047                               15BA
40 VPOKE 14336+F,PEEK(EN!+F)                     1C02
50 NEXT                                          1D9B
```

Após rodar este programa, cada SPRITE corresponderá a um dos 256 caracteres do MSX, sendo seu número o próprio código ASC do caractere.

Para ver um dos mil efeitos possíveis a serem empregados com esse recurso, digite a complementação do programa, a seguir:

```
60 INPUT A$                                      2028
70 L=LEN(A$) : C=3                               2640
80 FOR I=1 TO L                                  2AC4
90 CH$=MID$(A$,I,1)                              3111
100 CH=ASC(CH$)                                  3B79
110 FOR Y=0 TO 96                                3FB4
120 PUT SPRITE I,((C+I)*8,Y-1),,CH               4F59
130 NEXT                                         5013
140 LOCATE C+I-2,12,0 : PRINT CH$                5F68
150 NEXT                                         600F
```

Para rodá-lo, se você já rodou o programa anterior, basta digitar GOTO 60. Desta forma não perdemos o tempo de transferência da tabela de caracteres para a área de SPRITES da VRAM. O programa pede a entrada de uma string (seu nome, por exemplo). Evite os caracteres gráficos de código entre 0 e 31.

Implemente agora seu programa de maneira a aceitar qualquer caractere.

2.Q - ARLEQUIM BÊBADO

Quando ativamos a SCREEN 1, os 32 bytes da VRAM compreendidos entre os endereços 8192 e 8223 ficam reservados para atributos de cores. Cada um desses bytes define a cor de frente e a cor de fundo de um conjunto de 8 caracteres (8*32=256).

Digite o programa a seguir para entender melhor este mecanismo.

```
100 SCREEN 1:KEY OFF:DEFINT A-Z                 619
110 FOR L=0 TO 15                                A1E
120 FOR C=0 TO 15                                E34
130 VPOKE 6182+32*L+C,16*L+C                    171B
140 NEXT C                                      1928
150 NEXT L                                      1B75
160 LOCATE 3,20,0:PRINT"BYTE ALTERADO=";        23AD
170 GOTO 190                                    27BD
180 A=STICK(0):IF A=0 THEN GOTO 180             32D3
190 I=I-(A=3)+(A=7)-2*(A=5)+2*(A=1)             45F4
200 COLOR 15,1,1                                4B72
210 VPOKE 8192+I,&B10111000                     57B2
220 LOCATE 17,20:PRINT 8192+I                   63BD
230 IF STICK(0)<>0 THEN GOTO 230                6ECB
240 GOTO 180                                    7321
```

A linha 100 configura o VDP para SCREEN 1, apaga as teclas de função e define todas as variáveis como inteiras para tornar o programa mais rápido.

As linhas de 110 a 150 colocam todos os caracteres na tela através do VPOKE para evitar problemas na impressão dos caracteres de controle (0 a 31 e 127) através da instrução PRINT.

A linha 210 insere, num dos 32 bytes citados, um valor que define a cor de frente e a cor de fundo segundo o seguinte critério: os 4 bits da esquerda definem a cor de frente (no nosso exemplo &B1011 = 11 = amarelo) e os 4 bits da direita a cor de fundo (&B1000 = 8 = vermelho).

Se você quiser "vpokear" um número em decimal, basta calculá-lo segundo a regra:

VALOR DECIMAL=(COR DE FRENTE)*16+(COR DE FUNDO)

No nosso exemplo, o valor decimal seria:

11*16 + 8 = 184

Experimente substituir esse valor na linha 210 e

depois invente outras combinações de cores.

A linha 190 permite alterar o endereço do VPOKE usando o recurso do "parênteses lógico".

Se a afirmação for verdadeira seu valor será -1, se for falsa, será 0. Dessa forma, pressionando as teclas de setas (STICK(0)), você pode alterar o valor de I e, portanto, o endereço do byte da VRAM a ser alterado.

Assim, por meio desse programa, você pode visualizar quais são os grupos de 8 caracteres que têm sua cor alterada em função do byte de atributo que você mudou.

Se você, por exemplo, alterar os bytes 8197, 8198 e 8199 com os comandos

```
VPOKE8197,184:VPOKE8198,184:VPOKE8199,184
```

e comandar LIST para um programa em BASIC na SCREEN 1, verá uma listagem colorida apenas nos algarismos e símbolos aritméticos.

Note que a alteração no byte que inclui o caractere "espaço vazio" colore toda a tela ao redor da tabela de caracteres (pois ela está cheia de espaços vazios) e que uma alteração no último dos 32 bytes de atributos de cor (8233) altera a cor do cursor (caractere de código 255), permitindo uma visualização mais fácil do mesmo na hora de editar um programa.

Apague as linhas de 160 a 240 com"DELETE 160-240" e acrescente estas linhas ao programa:

```
160 FOR I=1 TO 32
170 X=INT(RND(-TIME)*14+1)
171 Y=INT(RND(-TIME)*14+1)
180 IF X=Y THEN GOTO 170
190 VPOKE 8191+I,X*16+Y
200 NEXTI
```

Desta forma você estará sorteando um número de 1 a 15 para a cor de frente e fundo (a cor transparente foi eliminada) e verificando se elas são diferentes.

Rode o programa várias vezes para ver seu efeito e observe como fica uma listagem na SCREEN 1: seu programa fica parecendo um arlequim bêbado!

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.2 - páginas 18 e 19.

2.R - USANDO 40 OU 64 COLUNAS NA SCREEN 2

Normalmente a impressão de caracteres na SCREEN 2 é semelhante à da SCREEN 1, em 32 colunas. Entretanto, com um pequeno programa em Linguagem de Máquina pode-se fazer com que a impressão assemelhe-se à da SCREEN 0, em 40 colunas. Com ligeiras alterações nessa mesma rotina, podemos fazê-la imprimir em 64 colunas. É exatamente isso que faz o programa listado a seguir. Digite-o e, antes de mais nada, grave-o em disco ou em fita. Depois rode-o.

```
100 SCREEN 0 : WIDTH 40                          2EB
110 CLEAR 200,&HE000                             683
120 FOR F=&HE000 TO &HE0D1                       D40
130  READ A$ : POKE F,VAL("&H"+A$)               1476
140 NEXT F                                       1692
150 PRINT,,,,,,,,,,                              1C83
160 PRINT SPC(10);"[ 1 ] 40 COLUNAS"             2596
170 PRINT SPC(10);"[ 2 ] 64 COLUNAS"             3311
180 A$=INKEY$                                    3606
190 IF A$="1" THEN 230                           3EEA
200 IF A$<>"2" THEN 180                          4703
210  POKE &HE0AD,4                               48E2
220  POKE &HE0CA,255                             5103
230 SCREEN 0 : NEW                               579C
240 DATA FE,03,C0,3A,AF,FC,FE,02                 6BA3
250 DATA C0,EB,46,23,5E,23,56,04                 7EA2
260 DATA 05,C8,1A,CD,19,E0,13,18                 8413
270 DATA F7,F5,C5,D5,E5,FD,E5,ED                 89FE
280 DATA 4B,B7,FC,ED,5B,B9,FC,CD                 9087
290 DATA 39,E0,ED,43,B7,FC,ED,53                 98FE
300 DATA B9,FC,FD,E1,E1,D1,C1,F1                 A530
310 DATA C9,CD,AB,00,D0,20,07,FE                 B80D
320 DATA 0D,28,73,FE,20,D8,6F,26                 C982
330 DATA 00,29,29,29,C5,D5,ED,5B                 DFE5
340 DATA 20,F9,19,11,40,FC,06,08                 F511
350 DATA C5,D5,3A,1F,F9,CD,0C,00                 11D
360 DATA FB,D1,C1,12,13,23,10,F0                 65D
370 DATA D1,C1,3A,E9,F3,32,F2,F3                 C5A
380 DATA FD,21,40,FC,D5,26,08,CB                 158A
390 DATA 7A,20,2A,CD,BF,E0,38,2B                 1E27
400 DATA C5,2E,06,FD,7E,00,CB,78                 3162
410 DATA 20,15,CD,C8,E0,38,15,CB                 427A
420 DATA 7F,28,0C,F5,D5,E5,CD,11                 555F
430 DATA 01,CD,20,01,E1,D1,F1,07                 6A1C
440 DATA 03,2D,20,E2,C1,FD,23,13                 7DA5
450 DATA 25,20,CC,D1,21,06,00,09                 8589
460 DATA 44,4D,CD,C8,E0,D0,01,00                 8890
```

```
470 DATA 00,21,08,00,19,EB,C9,E5
480 DATA 21,BF,00,B7,ED,52,E1,C9
490 DATA E5,21,EF,00,B7,ED,42,E1
500 DATA C9,00,00,00,00,00,00,00
```

Com isso, a rotina de impressão não standard já estará carregada na memória do micro e pronta para ser usada.

Note que a impressão em 40 ou 64 colunas deve ser feita na SCREEN 2 usando-se a sintaxe:

```
A$=USR0("string a ser impressa")
```

O programa a seguir ilustra o uso da impressão em 40 colunas.

```
10 SCREEN 2
20 PRESET (0,10)
30 A$=USR0("Teste de 40 colunas")
40 OPEN "GRP:" AS #1
50 PRESET (0,20)
60 PRINT #1,"Teste de 32 colunas"
70 GOTO 70
```

A impressão normal, usando o PRINT # , continua a funcionar sem alterações, em 32 colunas.

Observe também que quando se usa a opção de 64 colunas os caracteres normais se sobrepõem. Isso pode ser evitado se você redefinir uma fonte de caracteres (veja a dica 2.8) apropriadamente e rodar o programa a seguir.

```
100 DEFUSR=&HE000
110 POKE &HF91F,2
120 POKE &HF920,&H80
130 POKE &HF921,&HBB
140 SCREEN 2
150 PRESET(0,10)
160 A$=USR("123456789_123456789_12345678
9_123456789_123456789_123456789_1234")
170 LINE (0,18)-(255,18),15
180 GOTO 180
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.5 - CARIMBANDO SPRITES 8 x 8 NA SCREEN 2

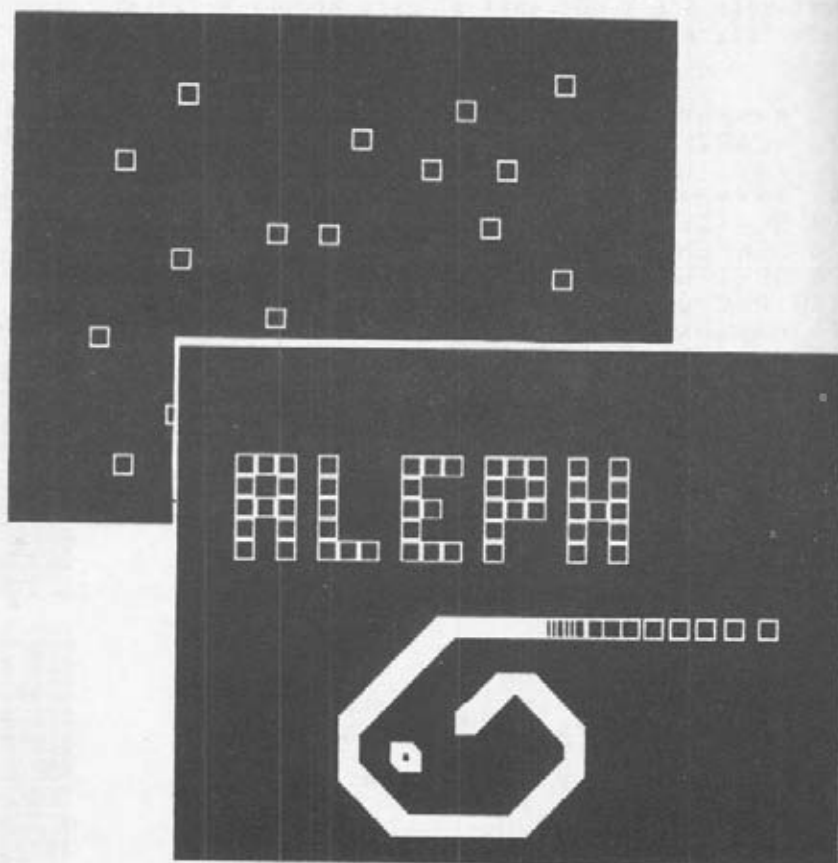
O recurso dos SPRITES é muito útil na apresentação visual dos programas, mas eles apresentam algumas limitações, como por exemplo o fato de 5 sprites não serem impressos pelo VDP se estiverem na mesma linha.

A rotina apresentada a partir da linha 120 na listagem abaixo imprime o SPRITE 8 x 8 cujo padrão é definido pela variável SC% na coordenada dada pelas variáveis X e Y. O canal #1 para arquivos também deve estar livre para ser utilizado pela rotina.

```
1 '********************                          520
2 '*CARIMBADOR 8x8    *                          C21
3 '*BY THE DOCTOR LUZ*                           1230
4 '********************                         108E
10 SL=(INP(&HA8)AND&HC0)/64                     2713
20 SCREEN 2,0:SC%=0                             20B5
30 SPRITE$(SC%)=CHR$(255)+STRING$(6,CHR$        4408
(&B10000001))+CHR$(255)
40 A$=INKEY$                                    477F
50 PUTSPRITE 10,(X,Y),15,SC%                    5509
60 IF A$=CHR$(28)THEN X=X+1                     60E1
70 IF A$=CHR$(29)THEN X=X-1                     6C9F
80 IF A$=CHR$(30)THEN Y=Y-1                     7904
90 IF A$=CHR$(31)THEN Y=Y+1                     80E0
100 IF A$="F"THEN SCREEN 0:END                  8759
110 IF A$=" " THEN GOSUB 130                    8CF2
120 GOTO 40                                     8F10
130 FORL% =0 TO 7                               9475
140 POKE &HC200+L%,VPEEK(BASE(14)+L%+8*S        A112
C%)
150 NEXT                                        A369
160 POKE &HF91F,SL                              A80F
170 POKE &HF920,0                               ACCF
180 POKE &HF921,&HC0                            B1F1
190 OPEN "GRP:" AS #1                           BC5F
200 PRESET (X,Y+1):PRINT#1,"a"                  C7B6
210 POKE &HF91F,&H0                             CE1A
220 POKE &HF920,&HBF                            D481
230 POKE &HF921,&H1B                            DB08
240 CLOSE #1                                    DFE0
250 RETURN                                      E139

TOTAL =                                         E139
```

A idéia desse carimbador de SPRITES é muito simples. A variável SC% ajuda a encontrar na VRAM o início da tabela de formação do SPRITE, que é transferido para a RAM a partir do endereço &HC200.

Em seguida, muda-se o conteúdo da variável do sistema CGPNT (em &HF91F) para que ao mandarmos imprimir o caracter "@" na tela gráfica, não seja impressa a matriz de pontos da ROM, mas o padrão do SPRITE que foi transferido a partir do endereço &HC200.

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.T - CARIMBADOR DE SPRITES 16 x 16 NA SCREEN 2

O princípio de funcionamento do programa a seguir é o mesmo que o do carimbador de SPRITES 8 x 8, apresentado na dica anterior. Contudo, agora a transferência da VRAM para RAM é um pouco mais demorada, pois os SPRITES 16 x 16 são definidos por 32 bytes cada um .

```
10 '********************
20 '*CARIMBADOR 16x16 *
30 '*BY THE DOCTOR LUZ*
40 '********************
50 SL=(INP(&HA8)AND&HC0)/64
60 SCREEN 2,2:SC%=1
70 SPRITE$(SC%)=CHR$(255)+STRING$(14,CHR
$(&B10000000))+CHR$(255)+CHR$(255)+STRIN
G$(14,CHR$(&B00000001))+CHR$(255)
80 A$=INKEY$
90 PUTSPRITE 10,(X,Y),15,SC%
100 IF A$=CHR$(28)THEN X=X+1
110 IF A$=CHR$(29)THEN X=X-1
120 IF A$=CHR$(30)THEN Y=Y-1
130 IF A$=CHR$(31)THEN Y=Y+1
140 IF A$="F"THEN SCREEN 0:END
150 IF A$=" " THEN GOSUB 170
160 GOTO 80
170 FORL% =0 TO 31
180 POKE &HC200+L%,VPEEK(BASE(14)+L%+32*
SC%)
190 NEXT
200 POKE &HF91F,SL
210 POKE &HF920,0
220 POKE &HF921,&HC0
230 OPEN "GRP:" AS #1
240 PRESET (X,Y+1):PRINT#1,"@B"
250 PRESET (X,Y+9):PRINT#1,"AC"
260 POKE &HF91F,&H0
270 POKE &HF920,&HBF
280 POKE &HF921,&H1B
290 CLOSE #1
300 RETURN
```

TOTAL = 85F

BIBLIOGRAFIA RECOMENDADA.

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.4 - ARMAZENANDO TELAS NA RAM

O programa em BASIC listado a seguir gera na memória do micro, a partir do endereço &HE000, uma rotina em Linguagem de Máquina que permite o armazenamento da tela na RAM e sua posterior recuperação, de volta para a VRAM.

A primeira parte do programa é a responsável pela geração da rotina de transferência. A segunda parte é um exemplo de como a rotina em L.M. deve ser usada.

Observe a incrível velocidade com que a rotina de transferência consegue armazenar (e recuperar!) os 16 Kbytes da VRAM !!! Em BASIC, essa mesma transferência, se fosse possível, certamente demoraria algo em torno de 100 vezes mais!

```
100 FOR F=&HE000 TO &HE03E                              57C
110   READ A$ : POKE F,VAL("&H"+A$)                      D01
120 NEXT F                                              EA2
130 DEFUSR0=&HE000 : DEFUSR1=&HE011                     173E
140 DATA CD,22,E0,21,00,00,11,00                        224E
150 DATA 40,01,FF,3F,CD,59,00,18                        2AA0
160 DATA 24,CD,22,E0,21,00,40,11                        3B82
170 DATA 00,00,01,FF,3F,CD,5C,00                        4D45
180 DATA 18,13,F3,DB,A8,47,CB,3F                        5F8E
190 DATA CB,3F,CB,3F,CB,3F,80,E6                        73A0
200 DATA FC,D3,A8,FB,C9,F3,DB,A8                        8773
210 DATA E6,F0,D3,A8,FB,C9,A2,00                        8D63
220 REM -----Exemplo de utilização-----                 96D8
230 COLOR 1,7,5 : SCREEN 2                              9D46
240 FOR F=80 TO 1 STEP -10                              A543
250   CIRCLE(128,80),80,1,,,80/F                        AE5C
260   CIRCLE(128,80),80,1,,,F/80                        BCA2
270 NEXT F                                              BF48
280 LINE (128,160)-(128,0)                              C8C1
290 LINE (48,80)-(208,80)                               D387
300 POKE 0,USR0(0) : ' VRAM para RAM                    E871
310 SCREEN 0 : WIDTH 38 : LOCATE 10,10                  F9A2
320 PRINT "Digite RETURN!":A$=INPUT$(1)                 986
330 SCREEN 2                                            AE3
340 POKE 0,USR1(0) : ' RAM para VRAM                    1410
350 GOTO 350                                            1663
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulos 0, 3 e 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.1 - páginas 31 a 54.

2.V - USANDO A VRAM PARA DADOS

O vídeo do MSX é controlado por um circuito especial chamado VDP, que tem a sua disposição 16 Kbytes de memória chamada Vídeo-RAM (VRAM).

O VDP pode usar a VRAM de quatro formas diferentes, correspondentes às SCREEN 0, 1, 2 e 3 do MSX.

Para cada SCREEN, a VRAM é dividida de forma diferente e sempre sobram algumas áreas não ocupadas. A seguir, apresentamos uma tabela onde relacionamos a área não usada da VRAM para cada SCREEN do MSX.

SCREEN	ÁREA NÃO USADA
0	960 até 2047 4096 até 16383
1	2048 até 6143 7040 até 8191 8224 até 14335
2	7040 até 8191
3	3584 até 6912 7940 até 14335

Todas as áreas livres da VRAM podem ser usadas para armazenamento de dados. Isso pode ser feito tanto com rotinas em Linguagem de Máquina como com o comando VPOKE do BASIC.

Vamos ilustrar de forma bem simples como se pode usar a VRAM para dados. Digite e execute o programa a seguir.

```
10 SCREEN 0 : WIDTH 38                        297
20 A$="DADOS EXEMPLOS PARA A VRAM"            CE3
30 VPOKE 4096,LEN(A$)                         110D
40 FOR F=1 TO LEN(A$)                         1768
50   VPOKE 4096+F,ASC(MID$(A$,F,1))           226D
60 NEXT F                                     248E
```

Com isso os dados da string **A$** estarão na VRAM. Agora você pode comandar NEW, apagando o programa e as

variáveis da memória RAM normal do micro.

Para recuperar os dados que foram passados para a VRAM, digite e rode o programa abaixo.

```
10 SCREEN 0 : WIDTH 39                 298
20 X=VPEEK(4096)                       629
30 A$=""                               858
40 FOR F=1 TO X                        C38
50    A$=A$+CHR$(VPEEK(4096+F))        1482
60 NEXT F                              1668
70 PRINT A$                            1868
```

Com isso, A$ estará novamente com os dados que havíamos gravado na VRAM.

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 105 e 106.

2.W - PESQUISADOR DE DESENHOS

O programa apresentado adiante é um editor de caracteres para telas gráficas. Digite-o, confira-o e grave-o.Depois, rode o programa.

Ao ser executado ele permite que o usuário "vasculhe" a memória do micro a procura de um desenho para tela gráfica. Os dois terços superiores da tela são usados para mostrar os desenhos dos caracteres da região da memória pesquisada tal qual eles apareceriam na figura da tela gráfica. O terço inferior da tela é usado para mostrar um único caractere em destaque. Esse caractere pode ser redesenhado e corresponde à ampliação das posições indicadas por um pequeno quadrado no centro da tela:

Assim que o programa é carregado, as teclas de setas permitem mover a região da memória pesquisada. Lembre-se que de &H0000 a &H7FFF existe ROM e a partir de &H8000 existe RAM. As funções das teclas de setas estão resumidas abaixo.

▶ : pesquisa um byte para a frente;
SHIFT + ▶ : pesquisa 8 bytes para frente (1 caractere);
◀ : pesquisa um byte para tráz;
SHIFT + ◀ : pesquisa 8 bytes para tráz (1 caractere);
▲ : pesquisa 256 bytes para frente (1 linha);
SHIFT + ▲ : pesquisa 2048 bytes para frente (1/3 de tela);
▼ : pesquisa 256 bytes para tráz (1 linha);
SHIFT + ▼ : pesquisa 2048 bytes para tráz (1/3 de tela).

Uma vez escolhida a região de memória a ser pesquisada, pode-se editar o caractere em destaque. Para isso deve-se digitar a tecla SELECT. Ela permite a passagem do modo "seleção" para o modo "edição". Pressionando SELECT a primeira vez um pequeno cursor surgirá no caractere em destaque na parte inferior da tela. Pode-se movê-lo com as teclas de setas. Para alterar um ponto, basta pressionar a barra de espaços: se ele estiver apagado, será aceso e, se estiver aceso, será apagado.

Com o caractere já editado, pode-se retornar do modo de edição digitando mais uma vez a tecla SELECT.

Para sair do programa e retornar ao BASIC, basta estar fora do modo de edição e pressionar a barra de espaços.

Se, ao retornar ao BASIC, a tela ficar toda com uma só cor, use um comando:

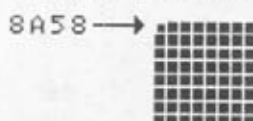
```
color 15,1,1
```

Isso deverá resolver o problema.

```
100 CLEAR 200,&HD000                                306
105 FOR F%=&HD000 TO &HD32A                         3ED
110   READ A$ : POKE F%,VAL("&H"+A$)                FF4
115 NEXT F%                                         11DE
120 BSAVE"ACHADES.BIN",&HD000,&HD32A                17CD
125 DEFUSR0=&HD000 : END                            1E13
130 DATA 3E,0F,32,EB,F3,CD,72,00                    26CC
135 DATA 3E,01,32,B0,FB,21,FA,D2                    3A36
140 DATA ED,5B,CF,F3,01,10,00,CD                    4BA3
145 DATA 5C,00,21,0A,D3,ED,5B,CD                    5C65
150 DATA F3,01,08,00,CD,5C,00,CD                    71A8
155 DATA B3,D1,E5,CD,07,D2,E1,2A                    8464
160 DATA F8,D2,11,00,00,01,00,10                    89F0
165 DATA CD,5C,00,06,01,21,E0,FB                    8F24
170 DATA 7E,CB,47,20,02,06,08,23                    994F
175 DATA 23,7E,CB,7F,20,0C,2A,F8                    A276
180 DATA D2,23,10,FD,22,F8,D2,C3                    AFE0
185 DATA 27,D0,CB,67,20,0C,2A,F8                    C1E7
190 DATA D2,2B,10,FD,22,F8,D2,C3                    D90E
195 DATA 27,D0,CB,77,20,0F,2A,F8                    E813
200 DATA D2,11,00,01,19,10,FD,22                    FCCF
205 DATA F8,D2,C3,27,D0,CB,6F,20                    787
210 DATA 11,2A,F8,D2,AF,11,00,01                    D18
215 DATA ED,52,10,F8,22,F8,D2,C3                    13C7
220 DATA 27,D0,21,E2,FB,7E,CB,47                    1D64
225 DATA 20,01,C9,2B,7E,CB,77,C2                    261B
230 DATA 3B,D0,CD,C0,00,21,11,D3                    38FC
235 DATA 36,04,2B,2B,3A,E2,FB,07                    4A49
240 DATA 30,03,CD,DB,D2,07,2B,30                    5160
245 DATA 03,CD,DB,D2,07,30,03,CD                    7210
250 DATA E1,D2,07,23,30,03,CD,E1                    841F
255 DATA D2,7E,FE,60,30,02,36,60                    8A5C
260 DATA FE,83,38,02,36,83,2B,7E                    8E36
265 DATA FE,8B,30,02,36,8B,FE,AE                    976F
270 DATA 38,02,36,AE,01,08,00,ED                    A0D0
275 DATA 5B,CD,F3,21,0A,D3,CD,5C                    AE3E
280 DATA 00,CD,E7,D2,21,E2,FB,7E                    C0DE
285 DATA 0F,DA,7E,D1,2A,F8,D2,01                    D843
290 DATA 80,08,09,3A,0E,D3,FE,90                    E9F1
```

```
295 DATA 38,01,23,FE,95,38,01,23       FF75
300 DATA FE,9A,38,01,23,FE,9F,38       9FF
305 DATA 01,23,FE,A4,38,01,23,FE       EE2
310 DATA A9,38,01,23,FE,AE,38,01       14B1
315 DATA 23,FE,B3,38,01,23,06,00       1F2A
320 DATA 3A,0F,D3,FE,60,20,04,CB       2720
325 DATA F8,18,36,FE,65,20,04,CB       38F8
330 DATA F0,18,2E,FE,6A,20,04,CB       4AE6
335 DATA E8,18,26,FE,6F,20,04,CB       5E33
340 DATA E0,18,1E,FE,74,20,04,CB       7381
345 DATA D8,18,16,FE,79,20,04,CB       85A6
350 DATA D0,18,0E,FE,7E,20,04,CB       8C1F
355 DATA C8,18,06,FE,83,20,02,CB       904B
360 DATA C0,7E,A8,77,2A,F8,D2,01       99DC
365 DATA 80,08,09,CD,07,D2,21,E1       A2E1
370 DATA FB,7E,CB,77,C2,A5,D0,DD       B046
375 DATA 21,0E,D3,DD,36,00,8B,DD       C2ED
380 DATA 23,DD,36,00,60,DD,23,DD       D370
385 DATA 23,DD,36,00,00,CD,C0,00       E95E
390 DATA 01,08,00,ED,5B,CD,F3,21       FD9F
395 DATA 0A,D3,CD,5C,00,CD,E7,D2       7A4
400 DATA C3,3B,D0,21,12,D3,CD,0E       D61
405 DATA 4B,3E,0F,32,E9,F3,3E,28       1490
410 DATA 32,B7,FC,3E,8A,32,B9,FC       1E16
415 DATA 2A,F8,D2,01,80,08,09,E5       2714
420 DATA 7C,CD,C8,D2,CD,CD,D2,E1       3A25
425 DATA E5,7C,CD,CD,D2,E1,E5,7D       4C21
430 DATA CD,C8,D2,CD,CD,D2,E1,E5       5F7E
435 DATA 7D,CD,CD,D2,3E,01,CD,8D       736E
440 DATA 00,3E,57,CD,8D,00,3E,01       847F
445 DATA CD,8D,00,3E,57,CD,8D,00       8ACC
450 DATA 3E,CF,CD,8D,00,E1,C9,01       909E
455 DATA 08,00,09,E5,AF,ED,42,C5       9A6C
460 DATA 7E,21,E9,F3,36,01,07,30       A3CB
465 DATA 02,36,0F,21,B7,FC,36,60       B25D
470 DATA 21,B4,00,41,2B,2B,2B,2B       C3FE
475 DATA 2B,10,F9,22,B9,FC,08,3E       D56F
480 DATA D3,CD,8D,00,08,21,E9,F3       EA7E
485 DATA 36,01,07,30,02,36,0F,21       FF6D
490 DATA B7,FC,36,65,08,CD,8D,00       851
495 DATA 08,21,E9,F3,36,01,07,30       D1E
500 DATA 02,36,0F,21,B7,FC,36,6A       1399
505 DATA 08,CD,8D,00,08,21,E9,F3       1DF3
510 DATA 36,01,07,30,02,36,0F,21       279A
515 DATA B7,FC,36,6F,08,CD,8D,00       3960
520 DATA 08,21,E9,F3,36,01,07,30       493A
525 DATA 02,36,0F,21,B7,FC,36,74       510E
530 DATA 08,CD,8D,00,08,21,E9,F3       72A7
535 DATA 36,01,07,30,02,36,0F,21       8401
```

```
540 DATA B7,FC,36,79,08,CD,8D,00     8997
545 DATA 08,21,E9,F3,36,01,07,30     8C63
550 DATA 02,36,0F,21,B7,FC,36,7E     95F4
555 DATA 08,CD,8D,00,08,21,E9,F3     9EE5
560 DATA 36,01,07,30,02,36,0F,21     A06D
565 DATA B7,FC,36,83,08,CD,8D,00     BF8A
570 DATA 08,C1,E1,0D,C2,0B,D2,C9     D1F3
575 DATA 0F,0F,0F,0F,C9,E6,0F,C6     E797
580 DATA 30,FE,3A,38,02,C6,07,CD     FC15
585 DATA 8D,00,C9,35,35,35,35,35     491
590 DATA C9,34,34,34,34,34,C9,D9     8F5
595 DATA 21,29,D3,46,C5,06,00,C5     F5A
600 DATA C1,10,FC,C1,10,F6,D9,C9     181C
605 DATA 00,B0,FF,FF,C3,C3,C3,C3     2086
610 DATA FF,FF,F8,88,88,88,F8,00     3320
615 DATA 00,00,3F,80,00,06,8B,60     4337
620 DATA 01,00,28,34,30,2C,31,33     5809
625 DATA 38,29,F2,28,37,32,2C,31     6F00
630 DATA 34,36,29,2C,31,2C,42,46     7F50
635 DATA 00,32,55                    820A

                         TOTAL  =    820A
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.X - IMAGENS INSTANTÂNEAS

Muitas vezes a produção de uma tela por um programa em BASIC é demorada demais, principalmente se estamos trabalhando com a SCREEN 2.

Podemos fazer com que a imagem não seja mostrada durante a confecção do desenho, mas somente após ele estar pronto.

Uma forma bem simples de conseguir isso é usando o comando COLOR como exemplificamos a seguir.

```
100 COLOR 0,9,9 : SCREEN 2                              974
110 REM --------  tela exemplo --------                 AF6
120 FOR F=0 TO 125 STEP 10                              10F1
130   G=80-F*80/125                                     1604
140   LINE (F,80)-(125,G)                               1E52
150   LINE (125,G)-(250-F,80)                           26E3
160   LINE (F,80)-(125,160-G)                           33FA
170   LINE (125,160-G)-(250-F,80)                       426A
180 NEXT F                                              44E1
190 COLOR ,,1                                           48DC
200 BEEP : GOTO 200                                     4DEC
```

Na linha 100 fazemos com que a cor de frente, de fundo e da borda da tela sejam iguais, impossibilitando a visão do que está sendo feito na tela. A linha 190 só é executada após o desenho estar terminado e coloca as cores na tela de uma só vez. Dessa forma, não vemos o desenho ser feito.

Experimente executar o programa novamente eliminando o comando COLOR da linha 100. Você verá o desenho sendo elaborado na tela.

Uma outra forma de conseguir o mesmo efeito é usando as rotinas do BIOS DISSCR (&H41) e ENASCR (&H44) como no programa a seguir.

```
10 DEFUSR0=&H41 : DEFUSR1=&H44                          66A
20 POKE 0,USR0(0)                                       8CA
30 FOR F=1 TO 24                                        C5B
40   PRINT "XXXXXXXXXXXXXXXXXXXXXXXXX"                  1445
50 NEXT F                                               16C2
60 POKE 0,USR1(0)                                       189A
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - página 147.
Coleção de Programas para MSX v.2 - páginas 42 e 68.

2.Y - IMPRESSOR EM TAMANHO DUPLO NA SCREEN 2

Aqui vai um programinha em linguagem de máquina que permite ao seu programa em BASIC imprimir qualquer mensagem em tamanho dobrado, na SCREEN 2. A grande utilidade dele é a de preencher a lacuna existente entre as SCREEN's 2 e 3, já que na primeira impressão é em tamanho natural, e na segunda em letras tiranossáuricas.

A utilização do programa é simples: após carregar a rotina em L.M. na memória, basta chamá-la pela função USR("texto") ou USR (var$). O posicionamento na tela é feito pelo comando PSET, PRESET ou outro comando gráfico, e as cores de frente e fundo são dadas pelo comando COLOR.

```
1000 DATA 3A,63,F6,FE,03,C0,21,F8
1010 DATA F7,7E,23,66,6F,7E,32,E0
1020 DATA D0,23,7E,32,E1,D0,23,7E
1030 DATA 32,E2,D0,3A,B5,FC,32,DE
1040 DATA D0,06,08,C5,CD,2B,D0,C1
1050 DATA 10,F9,C9,0E,02,3A,E0,D0
1060 DATA 32,E3,D0,C5,3A,B3,FC,32
1070 DATA DD,D0,2A,E1,D0,7E,E5,26
1080 DATA 00,6F,29,29,29,11,BF,1B
1090 DATA 19,3E,08,90,C5,4F,06,00
1100 DATA 09,7E,06,06,17,CD,6F,D0
1110 DATA 10,FA,C1,E1,23,3A,E3,D0
1120 DATA 3D,32,E3,D0,20,D7,21,DE
1130 DATA D0,34,C1,0D,20,BF,C9,21
1140 DATA DF,D0,36,FF,38,02,36,00
1150 DATA F5,C5,CD,8B,D0,21,DD,D0
1160 DATA 34,CD,8B,D0,21,DD,D0,34
1170 DATA C1,F1,C9,3A,DE,D0,4F,E6
1180 DATA 07,6F,79,CB,3F,CB,3F,CB
1190 DATA 3F,67,3A,DD,D0,E6,F8,06
1200 DATA 00,4F,09,3A,DD,D0,E6,07
1210 DATA 47,3E,08,90,47,AF,37,17
1220 DATA 10,FD,47,3A,DF,D0,A7,28
1230 DATA 06,CD,4A,00,B0,18,07,78
1240 DATA 2F,47,CD,4A,00,A0,CD,4D
1250 DATA 00,3A,E9,F3,87,87,87,87
1260 DATA 47,3A,EA,F3,B0,01,00,20
1270 DATA 09,CD,4D,00,C9,FA,6A,00
1280 DATA 14,11,80,00,FIM
1290 CLS:FOR I=&HD000 TO &HD0E3:READ A$:
POKE I,VAL("&H"+A$):NEXT I:END
```

Checksums: 68A, 8A6, 148A, 1D29, 2942, 3B97, 4005, 610F, 77A3, 83F7, 897F, 8F87, 9906, A0E9, B27C, C51A, D8A4, ECE8, 8A, 706, BC7, 14BE, 1DAE, 2A22, 3C83, 4C0B, 61A2, 7598, 8033, 8EF1

TOTAL = 8EF1

O próximo programa é um exemplo de utilização da rotina:

```
10 SCREEN 2                                   100
20 PRESET (10,80):COLOR 11,4                  482
30 DEFUSR=&HD000                              7E5
40 A$=USR("Milton Maldonado Jr.")             1080
50 COLOR 15,1,1                               1420
60 GOTO 60                                    170C
```

Dica especial: para mudar a altura das letras, use POKE &HD02C,n com n variando de 1 a 8. Que acontece quando n vale 0?

```
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
```

DICAS SONORAS

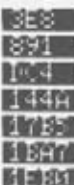
Neste capítulo abordamos vários recursos sonoros disponíveis nos micros MSX.

Além dos métodos normais, usando o PLAY e o SOUND, comentam-se alguns métodos não standard de geração de sons.

3.1 - Produzindo sons com ecos 87
3.2 - Percussão com o PLAY 88
3.3 - PLAY com SOUND 89
3.4 - Percussão com o click do teclado 90
3.5 - Teclado piano 91
3.6 - Editor de sons 95
3.7 - Músicas com programas 97
3.8 - Música aleatória 99
3.9 - Partitura sonora 100
3.A - Despertador 102
3.B - Digitalizador de voz 103

3.1 - PRODUZINDO SONS COM ECOS

A produção de sons com ecos nos micros MSX é extremamente simples através do comando PLAY. Para isso basta gerar um mesmo som nos três canais, porém com um pequeno atraso um em relação aos outros.

Veja o programa a seguir.

```
10 PLAY "S0M5000"
20 PLAY "C#32","R8C#32","R16C#32"
```

Ele gera um som com eco.

Uma outra forma é usar um envelope periódico. Por exemplo, digite e rode o programinha listado a seguir.

```
10 PLAY"T240S0M1000"
20 PLAY"O2CEGAA+AGE"
30 PLAY"O2CEGAA+AGE"
40 PLAY"FAO3CDD+DCO2A"
50 PLAY"O2CEGAA+AGE"
60 PLAY"O2DEFF+GFED"
70 GOTO 20
```

Agora altere a linha 10, deixando-a como mostramos abaixo, e execute o programa novamente.

```
10 PLAY"T240S8M1000"
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 119 a 121.
Curso de Basic MSX v.1 - aula 8.
Aprofundando-se no MSX - capítulo 5.

3.2 - PERCUSSÃO COM PLAY

A versatilidade do comando PLAY permite facilmente a geração de sons de percussão. Veja o exemplo a seguir:

```
10 PLAY "T120S0M9004L4AA06L16AAAAA4"
20 GOTO 10
```

Note que usamos o envelope 0 com um período de modulação muito curto. Experimente agora substituir T120 por T140 e M90 por M200.

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 119 a 121.
Curso de Basic MSX v.1 - aula 8.
Aprofundando-se no MSX - capítulo 5.

3.3 - PLAY COM SOUND

Uma outra forma de gerar sons de percussão com o comando PLAY é selecionando ruído para um ou mais canais através de um comando SOUND.

Experimente rodar o seguinte programa.

```
100 C$="O2S0M1500L8T136CCM3500CM1500CCCM
3500CM1500C
110 SOUND 6,32
120 FOR I=1 TO 2
130   SOUND 7,28
140   PLAY "","",C$
150 NEXT
160 C$="S0M10000C1...."
170 PLAY"","",C$
```

Altere o valor do registro 6 (sempre entre 1 e 32) e também a duração das notas e o tempo de execução. Você verá que pode produzir os sons mais variados. Um último recurso é mudar o formato da envoltória. Experimente alterá-lo, por exemplo, para S13. Agora, ao invés de uma "bateria", você deverá obter um "chocalho".

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 119 a 121 e 144 a 150.
Curso de Basic MSX v.1 - aula 8.
Aprofundando-se no MSX - capítulo 5.

3.4 - PERCUSSÃO COM O CLICK DO TECLADO

Uma terceira e última forma de produzir sons de percussão no MSX é usando o click do teclado. Digite e execute o programa a seguir.

```
10 KEY(1)ON:KEY(2)ON                          3CE
20 ON KEY GOSUB 70,80                         78D
30 X%=1                                       92B
40 OUT 170,127                                B3E
50 OUT 170,255                                E49
60 FOR F%=1 TO X%:NEXT F%:GOTO 40             1780
70 X%=X%+1 : RETURN                           1C71
80 X%=X%-1 AND X%>0 : RETURN                  2384
```

Use as teclas F1 e F2 para perceber o funcionamento do programa.

Esse recurso é usado de forma magistral no conhecido "joguinho" PITFALL. Além de música nos três canais ele usa o click do teclado para fazer a percussão.

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.2 - página 15.
Aprofundando-se no MSX - página 87.

3.5 - TECLADO PIANO

Existe uma maneira rápida e não muito sofisticada de se transformar um trecho do teclado de seu MSX num teclado musical.

No exemplo listado a seguir, as duas filas inferiores de teclas do MSX serão utilizadas.

Estas teclas correspondem ao Expert 1.1. No Hotbit haverá uma pequena modificação que será comentada adiante.

Vamos atribuir a cada uma das letras (ou símbolos) dessas duas filas o papel de uma tecla branca ou preta, conforme o esquema a seguir:

Para isto basta digitar o programa listado adiante.

```
10 SCREEN0:WIDTH39:KEY OFF:POKE&HFCAB,1   606
20 DIM N(255):PLAY"S0M2000L8"             C63
30 FOR I=0 TO 18                          1036
40 READ D$:D=ASC(D$):N(D)=I+41            1997
50 NEXT I                                 18C2
60 DATA Z,S,X,D,C,V,G,B,H,N,J,M,",",L,.,  28AB
Ç,;,/,*
70 A$=INPUT$(1):A=ASC(A$):N$="N"+STR$(N(  3000
A))
80 PRINT A$,:PRINT N$:PLAY N$             4128
90 GOTO 70                                400F

                     TOTAL  =             400F
```

Os que tiverem um Hotbit devem alterar a linha 60 para:

60 DATA Z,S,X,D,C,V,G,B,H,N,J,M,",",L,.,Ç,/,>,[

Nesta linha são definidas as teclas que fazem, em sequência, o papel das teclas de um piano. Você pode alterar esta linha para usar outro trecho do teclado do MSX. Lembre-se, porém, de ajustar o valor final de I no laço que vai de 30 a 50.

Na linha 40 o comando N(D)=I+36 ajusta a escala musical de maneira a fazer a primeira tecla (no nosso caso o Z) tocar o Dó central (N36).

Você também pode ajustar este valor de maneira a pegar outros trechos da escala musical em seu teclado.

Se você quiser alterar o timbre do seu "piano", modifique o PLAY da linha 20.

Como alteração final, sugerimos alguns acréscimos nas linhas 20, 70 e 80 de maneira a ficarem assim:

```
10 SCREEN0:WIDTH39:KEY OFF:POKE&HFCAB,1    [illegible]
20 DIM N(255):PLAY"S0M2000L8","S0M2000L8   [illegible]
","S0M2000L8"
30 FOR I=0 TO 18                           [illegible]
40 READ D$:D=ASC(D$):N(D)=I+36             199A
50 NEXT I                                  1BC5
60 DATA Z,S,X,D,C,V,G,B,H,N,J,M,",",L,.,   28AE
Ç,;,/,*
70 A$=INPUT$(1):A=ASC(A$):N$="N"+STR$(N(   [illegible]
A)):NA$="N"+STR$(N(A)+4):NB$="N"+STR$(N(
A)+7)
80 PRINT A$,:PRINT N$:IF N(A)=0 THEN GOT   [illegible]
O 70 ELSE PLAY N$,NA$,NB$
90 GOTO 70                                 4AB5

                              TOTAL =      4AB5
```

Experimente, ainda, alterar o comando SCREEN na linha 10 para:

SCREEN0,,0

Você deve estar curioso para saber como foram geradas as figuras que ilustram o começo desta dica. Para fazer isto, basta digitar este complemento do programa:

```
1000 OPEN"GRP:"AS #1
1010 COLOR 1,15,15:SCREEN 2
1020 T$="C1S4R10E2D14H2L10G2U14F2D10G2R1
4H2U10E2L14"
```

```
1030 FOR I= 1 TO 12
1040 READ A$
1050 PRESET(20*I,100):DRAW T$
1060 PRESET(20*I+3,102):PRINT#1,A$
1070 NEXT I
1080 DATA A,S,D,F,G,H,J,K,L,Ç,~,*
1090 FOR I= 1 TO 11
1100 READ A$
1110 PRESET(20*I+10,119):DRAW T$
1120 PRESET(20*I+13,121):PRINT#1,A$
1130 NEXT I
1140 DATA Z,X,C,V,B,N,M,",",.,;,/
1150 T1$="C1L20U48R9U36L18D36R9"
1160 T2$="C1L20U48BU18U18"
1170 FOR I=0 TO 11
1180 PRESET (45+20*I,162):IF I=0 OR I=3
OR I=7 OR I=10 THEN DRAW T2$ ELSE DRAW T
1$:PAINT(25+20*I,90),1,1
1190 NEXT I
1200 RETURN
```

Além disso você deve alterar as linhas 10 e 80 do programa original para:

```
10 POKE &HFCAB,1
80 IF N(A)=0 THEN GOTO 70 ELSE PLAY N$,N
A$,NB$
```

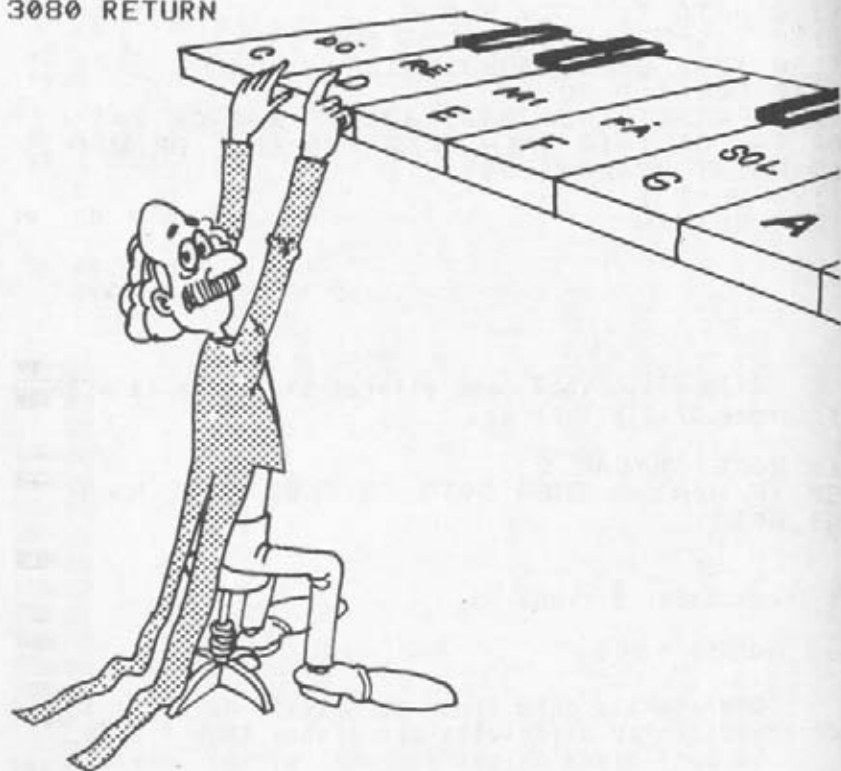
e acrescentar a linha 68:

```
68 GOSUB 1000
```

Obviamente, quem tiver um Hotbit deve fazer as correspondentes alterações nas linhas 1080 e 1140.

Se você ainda quiser dar uma última sofisticada em seu programa, acrescente-lhe as linhas a seguir:

```
25 GOSUB 3000
80 IF N(A)=0 THEN GOTO 70 ELSE PLAY N$,N
A$,NB$:PUT SPRITE 0,(21+10*X(P),139-19*Y
(P)),8,1
1015 FOR S=1 TO 8
1016 IF S=8 THEN S$=S$+CHR$(127) ELSE S$
=S$+CHR$(65)
```

```
1017 NEXT S
1018 SPRITE$(1)=S$
3000 DIM X(19),Y(19)
3010 X(0)=1
3020 FOR P=0 TO 18
3030 R=PMOD12
3040 X(P+1)=X(P)+1:IF R=4 OR R=11 THEN X
(P+1)=X(P)+2
3050 IFX(P)MOD2=1 THEN Y(P)=1 ELSE Y(P)=
2
3070 NEXTP
3080 RETURN
```

BIBLIOGRAFIA RECOMENDADA:

Curso de Basic MSX v.1 - aula 8.
Linguagem Basic MSX - páginas 119 a 121.

3.6 - EDITOR DE SONS

O programa a seguir é de grande utilidade na procura de valores corretos para serem usados com o comando SOUND.

Ao ser executado, ele apresenta uma tabela com os números dos registros do PSG à esquerda. Para selecionar um dado registro deve-se usar a tecla de seta para cima. O número do registro selecionado aparecerá na parte inferior da tela, logo abaixo da tabela. Para alterar seu valor deve-se usar as teclas de seta para a esquerda e seta para a direita. O registro 7 (de mixagem) é alterado de forma diferente: inicialmente deve ser selecionado com a tecla de seta para cima; depois deve-se pressionar a barra de espaços; a seguir, basta pressionar 8 teclas, 0 ou 1, para programar em binário o registro; após ter inserido os valores 0 e 1 nos bits do registro 7, pode-se pressionar RETURN para voltar ao modo de seleção normal do programa.

É recomendável que se conheça o funcionamento dos registros do PSG para programá-los conscientemente.

```
10 '********************
20 '* BY THE DOCTOR LUZ *
30 '********************
40 FOR L=&HC000 TO &HC145
50 READ A$:POKE L,VAL("&H"+A$)
60 NEXT
70 DEF USR=&HC000
80 PRINT"RODAR PROGRAMA ?"
90 A$=INPUT$(1)
100 IF A$="S" THEN A=USR(0) ELSE END
1000 DATA CD,C3,00,21,3C,C1,11,00
1010 DATA 00,01,0A,00,CD,5C,00,3E
1020 DATA 07,1E,38,CD,93,00,3E,08
1030 DATA 1E,0F,CD,93,00,3E,09,CD
1040 DATA 93,00,3E,0A,CD,93,00,26
1050 DATA 01,2E,02,CD,C6,00,CD,76
1060 DATA C0,CD,9F,00,FE,5A,C8,FE
1070 DATA 1C,28,0F,FE,1D,28,21,FE
1080 DATA 1E,28,65,FE,20,CA,C8,C0
1090 DATA 18,DD,DD,21,1B,C1,3A,2A
1100 DATA C1,16,00,5F,DD,19,DD,7E
1110 DATA 00,C6,01,DD,77,00,18,C7
1120 DATA DD,21,1B,C1,3A,2A,C1,16
1130 DATA 00,5F,DD,19,DD,7E,00,D6
1140 DATA 01,DD,77,00,18,B1,3A,2B
```

```
1150 DATA C1,DD,21,1B,C1,DD,77,07       2096
1160 DATA 06,0F,0E,00,59,CD,F6,C0       2AA5
1170 DATA 3E,20,CD,A2,00,DD,7E,00       3072
1180 DATA 5F,79,DD,23,0C,CD,93,00       4086
1190 DATA CD,F6,C0,3E,0A,CD,A2,00       6132
1200 DATA 3E,0D,CD,A2,00,10,DD,C9       758F
1210 DATA 3E,0A,CD,A2,00,3E,0D,CD       8500
1220 DATA A2,00,3A,2A,C1,3C,FE,0E       8B0A
1230 DATA D4,C5,C0,32,2A,C1,5F,CD       9222
1240 DATA F6,C0,C3,27,C0,3E,00,C9       9C11
1250 DATA 21,09,09,CD,C6,00,3E,4D       A53D
1260 DATA CD,A2,00,3E,49,CD,A2,00       B50A
1270 DATA 3E,58,CD,A2,00,06,08,CD       C687
1280 DATA 9F,00,CD,A2,00,CB,22,FE       D957
1290 DATA 30,28,02,CB,C2,10,F0,7A       EF0F
1300 DATA 32,2B,C1,C3,27,C0,7B,CB       40F
1310 DATA 3F,CB,3F,CB,3F,CB,3F,21       102F
1320 DATA 2C,C1,D5,5F,16,00,19,7E       110A
1330 DATA CD,A2,00,D1,21,2C,C1,7B       1990
1340 DATA E6,0F,5F,16,00,19,7E,CD       22F7
1350 DATA A2,00,C9,3D,C1,49,C1,FA       2C89
1360 DATA 29,C1,00,0F,0F,0F,00,00       3EF1
1370 DATA 00,00,00,00,30,31,32,33       40F9
1380 DATA 34,35,36,37,38,39,41,42       6400
1390 DATA 43,44,45,46,52,45,47,2E       7B96
1400 DATA 20,56,41,4C,4F,52             892C

                           TOTAL    =    892C
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 144 a 150.
Aprofundando-se no MSX - capítulo 5.
Linguagem de Máquina MSX - páginas 144 a 146.

3.7 - MÚSICAS COM PROGRAMAS

O programa abaixo ilustra um típico caso de multiprocessamento das máquinas MSX. Enquanto executa as tarefas de processamento normais de um micro, o MSX pode, simultaneamente, executar músicas. Apenas a título de exemplo, usamos um processamento bem simples, que consiste apenas em mostrar na região central da tela alguns números.

Note que a cada 1/54 segundos o processamento normal é interrompido para que o gerador de sons possa ser programado. Enquanto o gerador de sons executa a programação, o processamento normal continua.

```
100 SCREEN 0,,0:WIDTH 39:KEY OFF
110 INTERVAL ON : DEFINT F,G
120 DIM P$(34),Q$(34)
130 GOSUB 250:K=1
140 P$(1)="t200s0m7000"
150 Q$(1)="t200s0m7000"
160 PLAY P$(1),Q$(1)
170 ON INTERVAL=54 GOSUB 620
180 FOR F=1 TO 1000
190  LOCATE 8,10,0
200  PRINT USING"#######";F^2;K
210  FOR G=1 TO 100 : NEXT G
220 NEXT F
230 GOTO 180
240 '
250 P$(2)="r4o3":Q$(2)="r4o3"
260 P$(3)="a4o4c2d4":Q$(3)="r4L64ao4cea"
270 P$(4)="e4.f8e4":Q$(4)="o3ao4cea"
280 P$(5)="d2o3b4":Q$(5)="o3gbo4dg"
290 P$(6)="g4.a8b4":Q$(6)="o3gbo4dg"
300 P$(7)="o4c2o3a4":Q$(7)="o3ao4cea"
310 P$(8)="a4.g#8a4":Q$(8)="dfao5d"
320 P$(9)="b2g#4":Q$(9)="o4eg#bo5d"
330 '
340 P$(10)="e2a4":Q$(10)="o4eg#bo5d"
350 P$(11)="o4c2d4":Q$(11)="o3ao4cea"
360 P$(12)="e4.f8e4":Q$(12)="o3ao4cea"
370 P$(13)="d2o3b4":Q$(13)="o3gbo4dg"
380 P$(14)="g4.a8b4":Q$(14)="o3gbo4dg"
390 P$(15)="o4c4.o3b8a4":Q$(15)="o3ao4ce
a"
400 P$(16)="g#4.f#8g#4":Q$(16)="o4eg#bo5
d"
410 P$(17)="a2.":Q$(17)="o3ao4cea"
420 P$(18)="a2r4":Q$(18)="o3ao4cea"
```

```
430 '
440 P$(19)="o4g2.":Q$(19)="l4o4ceg"
450 P$(20)="g4.f8e4":Q$(20)="ceg"
460 P$(21)="d2o3b4":Q$(21)="o3gbo4d"
470 P$(22)="g4.a8b4":Q$(22)="o3gbo4d"
480 P$(23)="o4c2o3a4":Q$(23)="o3ao4ce"
490 P$(24)="a4.g#8a4":Q$(24)="o3ao4ce"
500 P$(25)="b2g#4":Q$(25)="eg#b"
510 P$(26)="e2.":Q$(26)="eg#b"
520 '
530 P$(27)="o4g2.":Q$(27)="O4CEG"
540 P$(28)="g4.f8e4":Q$(28)="ceg"
550 P$(29)="d2o3b4":Q$(29)="o3gbo4d"
560 P$(30)="g4.a8b4":Q$(30)="o3gbo4d"
570 P$(31)="o4c4.o3b8a4":Q$(31)="o3ao4ce
"
580 P$(32)="g#4.f#8g#4":Q$(32)="eg#b"
590 P$(33)="a2.a2R4":Q$(33)="o3ao4ceo3a2
"
600 RETURN
610 '
620 K=K+1
630 IF K=34 THEN K=2
640 PLAY P$(K),Q$(K)
650 RETURN

TOTAL = 9144
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 108, 178 e 179.

3.8 - MÚSICA ALEATÓRIA

Se você tiver curiosidade em saber como será a música do ano 3000, basta mandar seu MSX soltar a imaginação.

Digite o programa a seguir e ouça seu MSX "compondo". Quando você não aguentar mais, digite CONTROL + STOP.

Se você, porém, tiver paciência e curiosidade, ouça atentamente e verifique que há um certo padrão e algumas passagens geniais. Afinal a geração dos números aleatórios que estão sob a música é feita pelo micro segundo uma rígida regra matemática.

```
100 PLAY "S0M8000","S0M8000","S0M8000"
110 L$="L"+STR$(INT(RND(-TIME)*31)*2+2)
120 X$=L$+"N"+STR$(INT(RND(-TIME)*60))
130 Y$=L$+"N"+STR$(INT(RND(-TIME)*30+50)
)
140 Z$=L$+"N"+STR$(INT(RND(-TIME)*16+80)
)
150 PLAY X$,Y$,Z$
160 GOTO 110
```

A composição do exemplo intitula-se "A ORDEM DO CAOS" e o andamento é "ALLEGRO VIVACE CON UN PIZZICO DI PAZZIA". Se você quiser alterar as regras mude a linha 100 (tentando outros envelopes) e as linhas 120 a 140, mudando a distribuição das notas. Lembre-se, porém, que o argumento do N no PLAY não pode ultrapassar 96.

Lembrando o famoso exemplo do físico JAMES JEANS (2 macacos imortais acorrentados a um piano por toda a eternidade acabarão tocando uma sonata de BEETHOVEN), tenha paciência e fique esperando a obra prima do seu MSX!

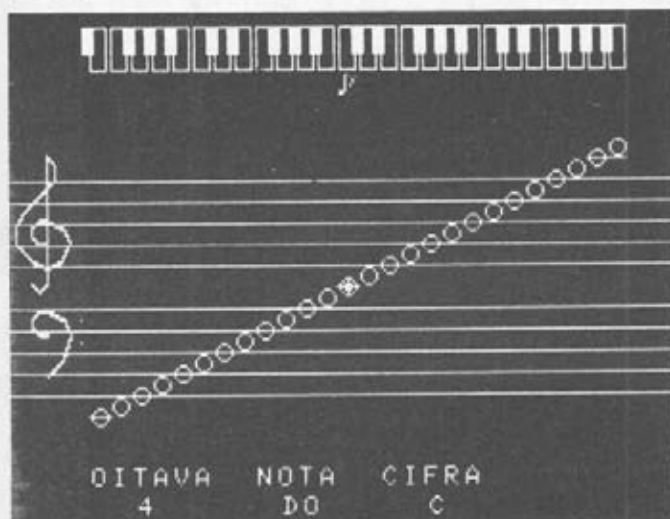
3.9 - PARTITURA SONORA

Excetuando-se alguns raros privilegiados que têm o chamado "ouvido absoluto", a maioria das pessoas tem uma certa dificuldade em identificar uma nota musical tocada individualmente.

Associar o som da nota a uma tecla do piano ou a uma posição na partitura musical se torna ainda mais difícil.

Para treinar seu ouvido e sua leitura de partitura, digite o programa a seguir. Para fazer a nota "subir" ou "descer" pela escala musical basta usar as teclas de seta para cima e seta para baixo. Uma vez escolhida a nota, basta apertar a barra de espaços para ouvir seu som.

```
100 P$(1)="DO      C":P$(2)="RE      D"
110 P$(3)="MI      E":P$(4)="FA      F"
120 P$(5)="SOL     G":P$(6)="LA      A"
130 P$(7)="SI      B"
140 COLOR 15,1,1:SCREEN 2,,0
150 OPEN"GRP:"AS #1
160 PRESET(31,170)
170 PRINT#1,"OITAVA  NOTA  CIFRA"
180 FOR C=0 TO 1
190 FOR L=63+48*C TO 95+48*C STEP 8
200 LINE(0,L)-(255,L)
210 NEXTL,C
220 FOR I=1 TO 26
230 D=23+8*I
240 PRESET(D,152-4*I):PRINT#1,CHR$(1)+
   CHR$(73)
250 LINE(D,5)-(D+6,21),,B
260 IF IMOD7=2 OR IMOD7=6 THEN 280
270 LINE(D-3,5)-(D+1,15),,BF
280 NEXT I
290 LINE(30,151)-(38,151):LINE(126,103)
   -(134,103):LINE(222,55)-(237,55)
300 D1$="C15S12LHUERM+2,+1M+1,+2M-1,+2M-
2,+1M-2,-1H2U2M+1,-2E4U2HD16GH"
310 PRESET(14,86):DRAW D1$
320 PRESET(14,87):DRAW D1$
330 D2$="C15S12LHUERM+2,+1M+1,+2M-1,+3G2
"
340 PRESET(14,120):DRAW D2$
350 PRESET(14,121):DRAW D2$:PSET(27,115)
:PSET(27,123)
```

```
360 FOR T=1 TO 8:C$=C$+CHR$(PEEK(7222+T)
):D$=D$+CHR$(PEEK(7206+T)):NEXT T:SPRITE
$(1)=C$:SPRITE$(2)=D$
370 I=13:PLAY"S0M7000"
380 A=STICK(0)
390 I=I+(A=5)-(A=1):X=23+8*I:Y=152-4*I
400 IF I<1 THEN I=1
410 IF I>26 THEN I=26
420 O=(I+15)\7:W=(I+15)MOD7+1
430 IF A=0 THEN 460
440 LINE (40,180)-(240,190),1,BF
450 PRESET(41,181):PRINT#1,O;"       ";P$(
W)
460 PUT SPRITE 3,(X,Y-1),,1
470 PUT SPRITE 4,(X,22),,2
480 N=12*(I\7)+14+2*(IMOD7)+(IMOD7>1)+(I
MOD7>5)
490 A$="N"+STR$(N)
500 IF STRIG(0) THEN GOSUB 520
510 GOTO 380
520 PLAY A$
530 IF STRIG(0) THEN 530
540 RETURN
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 119 a 121.
Curso de Basic MSX v.1 - aula 8.
Aprofundando-se no MSX - capítulo 5.

3.A - DESPERTADOR

Se você quiser usar seu MSX como relógio despertador, digite o programa a seguir e rode-o. Forneça a hora, minuto e segundo do momento de despertar e depois atualize o horário. Ao digitar o valor do SEGUNDO ATUAL digite um número um pouco maior e fique esperando até seu relógio de pulso indicar o mesmo número. Nesse instante pressione RETURN.

```
100 SCREEN 1:D=0:KEY OFF                              4EF
110 INPUT"HORA     DO DESPERTAR";HD                   DC6
120 INPUT"MINUTO   DO DESPERTAR";MD                   16C8
130 INPUT"SEGUNDO  DO DESPERTAR";SD                   1C5D
140 INPUT"HORA     ATUAL        ";H0                  252C
150 INPUT"MINUTO   ATUAL        ";M0                  3926
160 INPUT"SEGUNDO  ATUAL        ";S0                  4CA3
170 TIME=0                                            4EEA
180 CLS:LOCATE 3,10,0:PRINT"DESPERTADOR PARA  ";      6658
190 PRINT USING "##:##:##";HD;MD;SD                   7822
200 LOCATE 3,12:PRINT"    HORA CERTA  ";              8422
210 T=TIME\60:S1=(S0+T)MOD 60                         8C74
220 M=(S0+T)\60:M1=(M0+M) MOD 60                      9605
230 H=(M+M0)\60:H1=(H0+H) MOD 24                      A1F5
240 PRINT USING "##:##:##";H1;M1;S1                   AA60
250 D=(H1=HD)+(M1=MD)+(S1=SD)                         B858
260 IF D=-3 THEN PLAY "V15N40N42"                     C96A
270 IF STRIG(0) THEN D=0                              D167
280 GOTO 200                                          D597

                                      TOTAL  =        D597
```

Não esqueça de deixar o volume do micro ou da TV no máximo e siga o seguinte procedimento: vá dormir. Ao tocar o alarme, acorde! Levante e, rastejando, aproxime-se do micro. Pressione a barra de espaços para acabar com o maldito barulho e não volte para a cama! Se quiser um barulho mais irritante altere o PLAY da linha 260. Se chegar atrasado no serviço, leve a listagem do programa para seu chefe e tentem descobrir, juntos, onde você errou na digitação!

BIBLIOGRAFIA RECOMENDADA:

LINGUAGEM BASIC MSX - página 166.

3.8 - DIGITALIZADOR DE VOZ

Com este programa você pode digitalizar qualquer som e depois reproduzí-lo através do canal de áudio do micro. É muito útil em entradas de programas, que podem ser o trecho de uma música ou mesmo sua própria voz.

A entrada do som é feita pelo cabo de entrada (EAR) do cassete, e o som reproduzido sai pelo canal de áudio. Se o seu micro é um Expert, o som sairá pelo alto-falante interno. No Hot-Bit este sairá pela televisão.

A duração do som gravado é determinada pelos endereços &HC007 e &HC035. Os valores destes dois bytes devem ser iguais e compreendidos entre 0 e 170, e a maior duração é obtida com o valor 0. Para mudar a duração, basta dar POKEs nestes endereços com os valores acima especificados.

Você pode salvar uma informação digitalizada em disco ou fita com o comando:

```
BSAVE "NOME",ST,&HB000
```

onde

```
ST=256*PEEK(&HC007)
```

Naturalmente se o valor dado a &HC007 for menor que 128 o BSAVE não terá o efeito desejado, pois este salvará uma parte da ROM em vez de salvar a RAM abaixo de &H8000.

```
1000 REM -------------------                544
1010 REM Digitalizador de Voz               D19
1020 REM  By The Pilot  1988               1530
1030 REM -------------------               2006
1040 DATA F3,3E,AA,D3,A8,21,00,80          2800
1050 DATA 11,00,B0,AF,06,08,4F,DB          3A82
1060 DATA A2,E6,80,B1,CB,3F,00,00          4BEE
1070 DATA 00,00,00,00,00,00,10,EE          5EA1
1080 DATA 77,23,E5,ED,52,E1,38,E3          74A8
1090 DATA 3E,A0,D3,A8,FB,C9,F3,3E          882A
1100 DATA AA,D3,A8,21,00,80,11,00          8DAA
1110 DATA B0,7E,06,08,4F,1F,1F,D3          922F
1120 DATA AA,79,CB,3F,00,00,00,00          9BCC
1130 DATA 00,00,00,00,10,EE,23,E5          A556
1140 DATA ED,52,E1,38,E4,3E,A0,D3          B179
1150 DATA A8,FB,C9,FIM                     BCFE
```

```
1160 CLS:FOR I=&HC000 TO &HC05A:READ A$:D3AE
POKE I,VAL("&H"+A$):NEXT I
1170 PRINT "GRAVAR:DEFUSR=&HC000:PRINT UE0A8
SR(0)
1180 PRINT "TOCAR: DEFUSR=&HC02E:PRINT U208
SR(0)
1190 END                                  358
```

Se o seu micro for um Hotbit, mude a linha 1190 para:

```
1190 POKE &HC002,255:POKE &HC030,255:END
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 3.

DICAS PARA O CASSETE

Neste capítulo abordaremos o gravador cassete, seus recursos e suas utilizações.

Além dos recursos disponíveis normalmente no BASIC, várias aplicações somente possíveis através da Linguagem de Máquina são abordadas. O gravador pode ser usado com mais segurança e maior rapidez.

4.1 - Auto-execução de programas em cassete . 106
4.2 - Gravando programas BASIC em binário ... 108
4.3 - Gravando textos 112
4.4 - Lendo textos 113
4.5 - Verificando gravações 114
4.6 - Gravador de SCREEN 2 116

4.1 - AUTO-EXECUÇÃO DE PROGRAMAS EM CASSETE

O programa apresentado a seguir gera e grava em fita uma pequena rotina em Linguagem de Máquina capaz de carregar e auto-executar programas em BASIC gravados por CSAVE.

O procedimento para usá-lo é o seguinte:

- digite e execute o programa;
- prepare o gravador cassete com a fita em que será gravado o programa em BASIC;
- digite RETURN para gravar a rotina em L.M. (ela será gravada com o nome AUTOMA);
- digite NEW;
- digite o programa em BASIC a ser gravado em fita cassete;
- grave-o com o comando:

```
CSAVE"MATRIX"
```

Agora, a fita deve conter, em sequência, o programa AUTOMA (gravado por BSAVE) e o programa MATRIX (gravado por CSAVE). O programa AUTOMA pode ter qualquer outro nome, porém o programa MATRIX deve ser gravado exatamente com esse nome (com as letras maiúsculas!).

Para carregar e auto-executar o programa MATRIX, basta posicionar a fita no programa AUTOMA e comandar:

```
BLOAD"CAS:",R
```

Quando for carregado para a memória do micro o programa AUTOMA carregará e executará o programa MATRIX!

PROGRAMA GERADOR DO AUTOMA

```
100 REM                                 F3
110 REM AUTOCAS.BAS                     555
120 REM                                 66E
130 SCREEN 0                            83A
140 WIDTH 39                            9FF
150 CLEAR 200,&HC000                    DE7
160 FOR F=&HC000 TO &HC041              14C8
170   READ A$                           170A
180   A=VAL("&H"+A$)                    1E84
190   POKE F,A                          2206
200 NEXT F                              2549
210 PRINT,,,, "PREPARE O GRAVADOR E ";  2EF4
```

```
220 PRINT "PRESSIONE RETURN!"                    3F40
230 A$=INPUT$(1)                                 4539
240 BSAVE"CAS:AUTOMA",&HC000,&HC041              55E5
250 END                                          571C
260 REM                                          5801
270 REM DATA's para L.M.                         66E0
280 REM                                          681A
290 DATA F3,FD,21,B0,FB,FD,36,00                 7088
300 DATA 00,FD,36,01,01,21,34,C0                 8908
310 DATA 11,F0,FB,01,0D,00,ED,B0                 91B2
320 DATA 21,F0,FB,22,FA,F3,11,0E                 9735
330 DATA 00,19,22,F8,F3,21,2B,C0                 A148
340 DATA CD,3F,70,22,4D,41,54,52                 A913
350 DATA 49,58,22,00,43,4F,4C,4F                 B9B7
360 DATA 52,37,2C,31,3A,52,55,4E                 CAF0
370 DATA 0D,49,00,00,00,00,00,00                 DDE7
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 6.
Programação Avançada em MSX - capítulo 4.

4.2 - GRAVANDO PROGRAMAS BASIC EM BINÁRIO

Quando dispomos apenas do gravador cassete para armazenar dados de um programa, o processo de gravação e leitura se torna lento e pouco confiável. Uma forma de torná-lo mais prático é armazenando o programa e seus dados conjuntamente em formato binário. Para isso devemos dispor de um pequeno programa em Linguagem de Máquina que deve preparar a memória do micro para poder gravá-la sem perder os dados.

Vamos exemplificar o que acabamos de descrever. Imagine que você queira armazenar a matriz gerada pelo programa da figura a seguir.

```
100 DIM A$(5000,1)                          2BC
120 FOR F=0 TO 5000                         732
130  FOR G=0 TO 1                           BA0
140   A$(F,G)=STR$(2*F+G^2)                 1421
150  NEXT G                                 1690
160 NEXT F                                  18F3

                              TOTAL =       18F3
```

Uma das maneiras de fazer isso seria abrir um arquivo no cassete e gravar cada um dos dados nesse arquivo. Esse processo é extremamente demorado e portanto mais sujeito a falhas. Se pudermos gravar em binário (com BSAVE) a região da memória que contém as variáveis, e até o próprio programa, economizaremos tempo e aumentaremos a confiabilidade dos dados.

A seguir vamos apresentar um programa em BASIC que, ao ser executado, gera em fita ou em disco um programa em Linguagem de Máquina com o nome "BASBIN" (em fita) ou "BASBIN.BIN" (em disco). Antes de prosseguir, digite-o e execute-o. Depois, verifique se o programa BASBIN foi realmente gerado.

```
100 REM                                     F3
110 REM BINARIZADOR DE BASIC                80C
120 REM                                     91E
130 E=&HD000                                C54
140 READ A$ : A=VAL("&H"+A$)                1391
150 IF A$="XX" THEN 170                     1A30
160 POKE E,A : E=E+1 : GOTO 140             248F
170 E=&HD100                                2825
180 READ A$ : A=VAL("&H"+A$)                31B8
190 IF A$="XX" THEN 210                     3AEE
200 POKE E,A : E=E+1 : GOTO 180             480E
210 E=&HD200                                4C30
```

```
220 READ A$ : A=VAL("&H"+A$)                    5BC5
230 IF A$="XX" THEN 250                         634E
240 POKE E,A : E=E+1 : GOTO 220                 724F
250 BSAVE"BASBIN.BIN",&HD000,&HD28C             8186
260 END                                         8214
270 REM  DADOS EM &HD000                        8701
280 DATA F3,21,07,80,ED,4B,80,F3                8826
290 DATA CD,4C,D0,ED,4B,77,F1,CD                92E0
300 DATA 4C,D0,ED,4B,6A,F1,CD,4C                980A
310 DATA D0,ED,4B,68,F1,CD,4C,D0                A705
320 DATA ED,4B,68,F1,CD,4C,D0,ED                BA54
330 DATA 4B,A0,F0,CD,4C,D0,ED,4B                C05F
340 DATA C6,F6,CD,4C,D0,ED,4B,C4                E1F4
350 DATA F6,CD,4C,D0,ED,4B,C2,F6                F5EF
360 DATA CD,4C,D0,ED,4B,76,F6,CD                258
370 DATA 4C,D0,FB,C9,71,23,70,23                789
380 DATA C9,48,50,45,4C,41,20,41                CCE
390 DATA 52,4F,54,49,44,45,4F,54                1842
400 DATA 41,4E,45,52,XX                         1EB2
410 REM  DADOS EM &HD100                        282E
420 DATA F3,21,07,80,01,80,F3,CD                3908
430 DATA 4D,D1,01,77,F1,CD,4D,D1                4AFA
440 DATA 01,6A,F1,CD,4D,D1,01,68                5F76
450 DATA F1,CD,4D,D1,01,68,F1,CD                735B
460 DATA 4D,D1,01,A0,F0,CD,4D,D1                808A
470 DATA 01,C6,F6,CD,4D,D1,01,C4                86E3
480 DATA F6,CD,4D,D1,01,C2,F6,CD                8CEA
490 DATA 4D,D1,01,76,F6,CD,4D,D1                9688
500 DATA 21,55,D1,11,07,80,01,14                9FBF
510 DATA 00,ED,B0,FB,C9,7E,02,23                B130
520 DATA 03,7E,02,23,C9,45,44,49                C1B3
530 DATA 54,4F,52,41,20,41,4C,45                D473
540 DATA 50,48,20,2D,20,31,39,38                EBFE
550 DATA 37,XX                                  F2B7
560 REM  DADOS EM &HD200                        BB
570 DATA F3,21,07,80,01,80,F3,CD                672
580 DATA 65,D2,01,77,F1,CD,65,D2                AD7
590 DATA 01,6A,F1,CD,65,D2,01,68                12EA
600 DATA F1,CD,65,D2,01,68,F1,CD                1B44
610 DATA 65,D2,01,A0,F0,CD,65,D2                2764
620 DATA 01,C6,F6,CD,65,D2,01,C4                3A02
630 DATA F6,CD,65,D2,01,C2,F6,CD                4C30
640 DATA 65,D2,01,76,F6,CD,65,D2                6209
650 DATA 21,78,D2,11,07,80,01,14                772D
660 DATA 00,ED,B0,21,6D,D2,11,F0                828B
670 DATA FB,01,0B,00,ED,B0,21,F0                88AB
680 DATA FB,22,FA,F3,11,0C,00,19                9028
690 DATA 22,F8,F3,FB,C9,7E,02,23                9A6D
700 DATA 03,7E,02,23,C9,47,4F,54                A305
```

```
710 DATA 4F,20,30,30,31,32,30,0D        B614
720 DATA 45,44,49,54,4F,52,41,20        C5FD
730 DATA 41,4C,45,50,48,20,2D,20        D9B9
740 DATA 31,39,38,37,XX                 E7FA
```

Uma vez com o programa BASBIN gravado, vamos testá-lo, ao mesmo tempo em que aproveitamos para aprender como ele deve ser usado. Para isso, limpe a memória com um NEW e digite o programa a seguir exatamente como ele está listado, sem nenhum espaço a menos ou a mais!

```
100 REM EDITORA ALEPH - 1987                     610
110 CLEAR 200,&HD000                             9CD
120 X$="EDITORA ALEPH - 1987"                    1172
130 PLAY"T240S8M1000"                            17AE
140 PLAY"O2CEGAA+AGE"                            1BE9
150 PLAY"O2CEGAA+AGE"                            203A
160 PLAY"FAO3CDD+DCO2A"                          25EC
170 PLAY"O2CEGAA+AGE"                            3095
180 PLAY"O2DEFF+GFED"                            3CD7
190 SCREEN 0:PRINT ,,,,"               ";X$      4A1C
200 GOTO 140                                     4DBE

                               TOTAL   =         4DBE
```

Após tê-lo digitado, execute-o, espere alguns segundos e pressione CONTROL+STOP para interrompê-lo. Vamos agora preparar a memória para poder ser gravada em binário. Digite o seguinte comando:

```
BLOAD"BASBIN.BIN",R
```

A seguir, vamos gravar a memória com o programa e com os dados. Para isso, comande:

```
BSAVE"TESTES.BIN",&H8000,&HD300,&HD200
```

Com isso o programa em BASIC e seus dados serão gravados em formato binário.

Para tornar a carregá-lo, basta comandar:

```
BLOAD"TESTES.BIN
```

E a seguir, comandar:

```
DEFUSR=&HD100:?USR(0)
```

Experimente fazer isso e depois comande:

```
PRINT X$
```

Você verá que o conteúdo de X$ ainda está presente.

Se tivéssemos comandado:

```
BLOAD"BASBIN.BIN",R
```

O programa seria carregado e começaria a rodar automaticamente a partir da linha 120. Essa é uma outra forma de fazer programas em BASIC se auto-executarem logo após a carga a partir de fitas cassete.

Você pode usar o BASBIN.BIN com qualquer programa em BASIC, desde que ele não ocupe a memória acima de &HD000. O programa em BASIC deverá começar sempre com as duas primeiras como mostradas a seguir:

```
100 REM EDITORA ALEPH - 1987
110 CLEAR 200,&HD000
```

A rigor, o primeiro parâmetro do CLEAR da linha 110 pode ser alterado, mas o segundo deve ser necessariamente menor ou igual a &HD000 !

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

4.3 -GRAVANDO TEXTOS

Este programinha transfere uma mensagem (em ASCII) para a fita cassete. Após a chamada, ele transfere para a fita os dados definidos após &HC016 até o primeiro byte &H00, retornando então ao interpretador BASIC.

```
1000 DATA CD,EA,00,21,16,C0,7E,A7          6C3
1010 DATA 28,08,E5,CD,ED,00,E1,23          C1E
1020 DATA 18,F4,CD,F0,00,C9                123D
1025 REM a seguir está a mensagem          192A
1030 DATA 4F,53,20,43,59,4C,4F,4E          2326
1040 DATA 49,4F,53,20,49,4E,56,41          34C5
1050 DATA 44,49,52,41,4F,20,41,20          43E2
1060 DATA 54,45,52,52,41,20,21,00          5A9D
1070 CLS:FOR I=&HC000 TO &HC035            6557
1080 READ A$:POKE I,VAL("&H"+A$):NEXT I:7898
END
                              TOTAL  =     7898
```

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 4.

4.4 - LENDO TEXTOS

Este programa faz a função oposta à da dica 4.3: lê a fita e transfere para a tela a informação lida. Devido à simplicidade deste sistema, caracteres de controle (CR, LF, etc) não são reconhecidos como tais e aparecem na tela como símbolos gráficos. A leitura termina na primeira ocorrência do byte &H00.

Para posicionar a informação na tela use os comandos

```
LOCATE C,L:PRINT ;
```

onde C e L são as coordenadas de impressão desejada.

```
1000 DATA CD,E1,00,CD,E4,00,A7,28                6HD
1010 DATA 04,D3,98,18,F6,CD,E7,00                B14
1020 DATA C9,FIM                                 F5H
1030 CLS:FOR I=&HC100 TO &HC110:READ A$:224H
POKE I,VAL("&H"+A$):NEXT I:END
                                     TOTAL = 224H
```

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 4.

4.5 - VERIFICANDO GRAVAÇÕES

Quantas vezes você já não deu socos no seu gravador por causa de um programa que não carregou? Para programas em BASIC ainda existe o comando "CLOAD?", mas para programas em binário nada foi implementado no BASIC MSX visando a verificação de gravações.

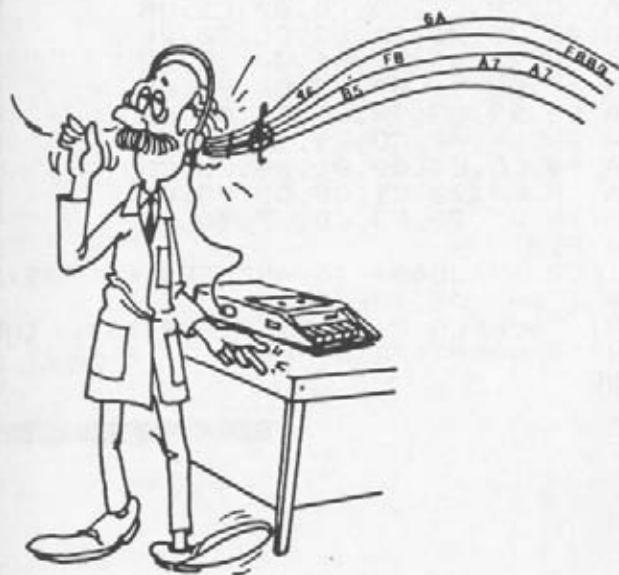
O programa a seguir instala na RAM uma sub-rotina em L.M. que faz a função de verificar um arquivo gravado no formato binário. Eis o programa:

```
1000 REM ----------------------------        6AB
1010 REM VERIFICADOR DE BSAVE V. 1.0        10F5
1020 REM  (C) 1987/88 BY THE PILOT          19F0
1030 REM ----------------------------       2447
1040 DATA CD,6C,00,AF,D3,99,D3,99           3884
1050 DATA CD,E1,00,38,76,06,0A,C5           4758
1060 DATA CD,E4,00,C1,10,F9,FE,D0           5AE8
1070 DATA 20,EE,06,06,C5,CD,E4,00           70AB
1080 DATA 38,61,D3,98,C1,10,F5,3E           84FD
1090 DATA 3E,D3,98,D3,98,CD,E1,00           8AEB
1100 DATA 21,57,F8,06,06,E5,C5,CD           8F41
1110 DATA E4,00,C1,E1,77,23,10,F5           9886
1120 DATA 3E,2C,2A,57,F8,CD,A9,D0           A0FD
1130 DATA D3,98,2A,59,F8,CD,A9,D0           AD38
1140 DATA D3,98,2A,5B,F8,CD,A9,D0           C031
1150 DATA 3E,20,06,12,D3,98,10,FC           D0D8
1160 DATA 2A,57,F8,ED,5B,59,F8,13           E620
1170 DATA AF,32,5D,F8,E5,D5,CD,E4           F9D3
1180 DATA 00,D1,E1,38,0E,BE,C4,A2           571
1190 DATA D0,23,A7,E5,ED,52,E1,20           865
1200 DATA EB,18,05,3E,01,32,5D,F8           10D5
1210 DATA 21,10,00,CD,C6,00,CD,E7           1B1F
1220 DATA 00,3A,5D,F8,A7,C0,21,FC           23B7
1230 DATA D0,7E,A7,C8,23,CD,A2,00           36AC
1240 DATA 18,F7,CD,A9,D0,CD,B4,D0           4884
1250 DATA C9,F5,7C,CD,C7,D0,7D,CD           5885
1260 DATA C7,D0,F1,C9,F5,3E,2D,D3           6FD8
1270 DATA 98,F1,CD,C7,D0,3E,20,D3           8248
1280 DATA 98,3E,01,32,5D,F8,C9,F5           87EB
1290 DATA CB,3F,CB,3F,CB,3F,CB,3F           8E7D
1300 DATA 01,EC,D0,81,4F,3E,00,88           96AE
1310 DATA 47,0A,D3,98,F1,E6,0F,01           9F85
1320 DATA EC,D0,81,4F,3E,00,88,47           AC6E
1330 DATA 0A,D3,98,C9,30,31,32,33           BE13
1340 DATA 34,35,36,37,38,39,41,42           CCE2
1350 DATA 43,44,45,46,0D,0A,53,45           E424
1360 DATA 4D,20,45,52,52,4F,53,0D           F948
```

```
1370 DATA 0A,00,FIM                              1DA
1380 CLS:PRINT "CARREGANDO ROTINA V-BSAV BFD
E"
1390 FOR I=&HD000 TO &HD109:READ A$:POKE 201B
 I,VAL("&H"+A$):NEXT I:PRINT "ROTINA CAR
REGADA.":END
```

Uso: após salvar o seu programa em binário, rebobine a fita e comande:

```
DEFUSR=&HD000:?USR(0)
```

Naturalmente, o seu programa não pode estar entre as posições &HD000 e &HD110 (por que?). A qualquer instante a verificação pode ser interrompida por CONTROL+STOP.

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 4.

4.6 - GRAVADOR DE SCREEN 2

Desta vez é um programinha que guarda a tela gráfica (SCREEN 2) em fita cassete, deixando de ser privilégio dos possuidores de disk drives essa operação.

Após carregar a rotina em L.M., basta chamar a rotina desejada pela função USR. Naturalmente, a função deve ser pré-programada, pois o computador apaga a SCREEN 2 no modo de comando.

```
1000 REM -------------------------           646
1010 REM COPIADOR SCREEN 2<-->FITA           1030
1020 REM JANEIRO 1988 - THE PILOT            1729
1030 REM -------------------------           2150
1040 DATA F3,DB,98,AF,D3,99,D3,99            304C
1050 DATA CD,EA,00,D8,CD,ED,00,CD            435C
1060 DATA ED,00,CD,ED,00,01,00,38            5676
1070 DATA DB,98,C5,CD,ED,00,C1,D8            6B7F
1080 DATA 0B,78,B1,20,F3,CD,F0,00            803E
1090 DATA C9,21,00,20,01,00,18,3E            8873
1100 DATA 1F,CD,56,00,F3,DB,98,AF            8E26
1110 DATA D3,99,D3,99,CD,E1,00,D8            9479
1120 DATA CD,E4,00,CD,E4,00,CD,E4            9CAE
1130 DATA 00,CD,E4,00,01,00,38,C5            A5EE
1140 DATA CD,E4,00,C1,D8,D3,98,0B            B89D
1150 DATA 78,B1,20,F3,CD,E7,00,C9            C965
1160 DATA FIM                                CEBC
1170 CLS:FOR I=&HD000 TO &HD05F:READ A$:E743
POKE I,VAL("&H"+A$):NEXT
1180 PRINT "SCREEN 2<-->FITA":PRINT:PRIN3A5
T "SALVAR: &HD000":PRINT"LER:    &HD029"
:PRINT:END

                         TOTAL    =     3A5
```

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 4.

5

DICAS PARA A IMPRESSORA

O acesso à impressora nos micros MSX pode ser realizado de várias maneiras diferentes, sendo que o usuário pode interferir em algumas delas.

As dicas deste capítulo desvendam aos usuários comuns recursos disponíveis somente através da Linguagem de Máquina ou de programas em BASIC bem construidos.

5.1 - Eco na impressora 118
5.2 - Transformando PRINT em LPRINT 119
5.3 - Impressão dupla 120
5.4 - HEXA-PRINTER e HEXA-SCREEN 122
5.5 - Filtro genérico 123
5.6 - Impedindo o uso da impressora 126
5.7 - Cópia gráfica 127
5.8 - Caracteres deitados 130
5.9 - Strings em modo gráfico 131
5.A - Máquina de escrever 132
5.B - Impressor de programas 133

5.1 - ECO NA IMPRESSORA

O MSX possui duas rotinas do BIOS para o envio de dados à impressora. Uma delas, a LPTOUT, pode ser facilmente usada para reproduzir o que for enviados para a tela diretamente na impressora. Uma aplicação desse tipo de recurso pode ser facilmente entendida se pensarmos em programas que apresentam resultados apenas na tela. Para fazê-los enviar os resultados para a impressora teríamos normalmente que alterá-los por inteiro. Obviamente, a maneira mais fácil é usar o programa apresentado a seguir.

```
100 CLEAR 200,&HE000                          316
110 FOR F=&HE000 TO &HE081                    991
120  READ A$ : POKE F,VAL("&H"+A$)           1065
130 NEXT F : DEFUSR0=&HE000                  165A
140 POKE 0,USR0(0) : END                     109D
150 REM                                      1F92
160 REM DADOS                                22A0
170 REM                                      24C5
1000 DATA FD,21,A4,FD,FD,36,00,C3            2002
1010 DATA FD,36,01,11,FD,36,02,E0            3EE5
1020 DATA C9,FE,7A,30,31,47,3A,80            4F92
1030 DATA E0,FE,01,78,28,3F,47,3A            63AA
1040 DATA 80,E0,FE,02,78,28,19,47            7A0B
1050 DATA 3A,80,E0,FE,03,78,28,49            8E2E
1060 DATA 47,3A,80,E0,FE,04,78,28            93A9
1070 DATA 07,FE,20,38,09,CD,A5,00            97C0
1080 DATA 21,80,E0,36,00,C9,FE,1B            9F4C
1090 DATA 28,0C,FE,0D,28,EF,FE,0A            A737
1100 DATA 28,EB,3E,20,18,E7,21,80            B363
1110 DATA E0,36,01,18,E8,FE,59,28            C558
1120 DATA 0A,FE,4E,28,0D,FE,4F,28            D754
1130 DATA 09,18,D5,21,80,E0,36,03            EEEF
1140 DATA 18,D3,21,80,E0,36,02,18             45A
1150 DATA CC,21,80,E0,36,04,18,C5             FE0
1160 DATA 00,00,52,45,4E,41,54,4F            140A

                               TOTAL   =     140A
```

Após executá-lo, pode-se apagá-lo da memória com o comando NEW. Entretanto é conveniente salvá-lo previamente em disco ou em fita.

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 6 e página 159.
Programação Avançada em MSX -capítulo 3 e página 149.

5.2 - TRANSFORMANDO PRINT EM LPRINT

O pequeno prògrama apresentado a seguir deve ser inserido no final de programas maiores. Para acioná-lo, deve-se comandar:

RUN 65000   (e RETURN)

Ao ser executado ele "varre" o programa principal a procura de instruções PRINT e as substitui por instruções LPRINT. Isso pode ser muito útil para redirecionar a saída de dados da tela para a impressora, porém o programa não deve ter instruções PRINT #, pois serão transformadas em LPRINT #, ocasionando erros de sintaxe. O programa também não deverá conter números cujo código compactado seja igual a token do PRINT (145), pois nesse caso eles terão seus valores alterados (observe como procedemos para evitar isso na linha 65010 do programa, ao invés de 145, escrevemos 100+45!).

```
65000 REM                                         274
65001 REM Muda PRINT p/ LPRINT                    B53
65002 REM                                         DBA
65003 EI = 32769!                                 12FD
65004 B1 = PEEK(EI)   : B2 = PEEK(EI+1)           1E5A
65005 B3 = PEEK(EI+2) : B4 = PEEK(EI+3)           28F7
65006 PL = B1 + 256*B2                            328A
65007 NL = B3 + 256*B4                            3C46
65008 IF B1=0 AND B2=0 THEN END                   4690
65009 FOR F=EI+4 TO PL-2                          5059
65010   IF PEEK(F)<>(100+45) THEN 65012           5DFF
65011      POKE F,157                             66FE
65012 NEXT F                                      693C
65013 EI=PL                                       6C39
65014 GOTO 65004                                  6E9B

                              TOTAL =             6E9B
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 2.

5.3 - IMPRESSÃO DUPLA

O programa apresentado a seguir gera uma rotina em Linguagem de Máquina a partir do endereço &HE000 capaz de fazer com que a impressão de listagens e outros dados seja feita com dupla passagem da cabeça de impressão. Com isso os textos ficarão mais legíveis e com maior contraste.

Após digitar e gravar o programa, você pode apagá-lo da memória com o comando NEW.

Caso você deseje desativar a dupla impressão, basta comandar:

```
POKE &HFFB6,&HC9
```

Para ativá-la novamente, uma vez que tenha sido desativada, basta comandar:

```
POKE &HFFB6,&HC3
```

Note que o programa em BASIC, ao ser rodado pede a quantidade de caracteres a serem impressos em cada linha. Uma vez especificado esse parâmetro, sempre que a dupla impressão estiver ativa, as linhas serão impressas com essa largura.

A rotina em Linguagem de Máquina não funcionará como esperado se o número de colunas especificado for maior que o número de colunas da impressora.

```
100 SCREEN 0 : CLEAR 200,&HE000
110 PRINT : PRINT : PRINT
120 PRINT"  Caracteres por linha ";
130 INPUT C
140 IF C<1 OR C>255 THEN RUN
150 FOR F=&HE000 TO &HE06F
160   READ A$:POKE F,VAL("&H"+A$)
170 NEXT F
180 FOR F=&HE070 TO &HE173
190   POKE F,0
200 NEXT F
210 POKE &HE02A,C
220 DEFUSR=&HE000 : POKE 0,USR0(0)
230 END
240 DATA F3,21,0E,E0,22,B7,FF,3E
250 DATA C3,32,B6,FF,FB,C9,F5,C5
260 DATA D5,E5,FE,0D,28,52,FE,0A
270 DATA 28,13,ED,4B,6F,E1,21,70
280 DATA E0,09,03,ED,43,6F,E1,77
290 DATA 79,FE,FF,20,3B,3E,C9,32
```

```
300 DATA B6,FF,CD,50,E0,CD,50,E0      H238
310 DATA ED,4B,6F,E1,78,B1,3E,0A      B235
320 DATA C4,A5,00,21,00,00,22,6F      C362
330 DATA E1,3E,C3,32,B6,FF,18,18      D6CF
340 DATA ED,4B,6F,E1,78,B1,C8,21      EC05
350 DATA 70,E0,7E,CD,A5,00,23,0D      AY
360 DATA 20,F8,3E,0D,CD,A5,00,C9      703
370 DATA E1,D1,C1,F1,33,33,B7,C9      CEE

                             TOTAL  =  CEE
```

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 3.

5.4 - HEXA-PRINTER E HEXA-SCREEN

Muitas vezes precisamos saber quais caracteres (normalmente de controle) um dado programa (normalmente em Linguagem de Máquina) está enviando para a impressora. Nesses casos, seria muito cômodo se ao invés de imprimir os caracteres propriamente ditos pudessémos imprimir seus códigos hexadecimais. É exatamente isso que o programa listado abaixo faz, com uma opção ainda mais útil para quem não possui impressora: a possibilidade de simular na tela do vídeo a saída de impressora. Digite e grave o programa. Depois rode-o e use algumas vezes o comando LPRINT.

```
100 FOR F=&HD000 TO &HD071                    5A8
110    READ A$:POKE F,VAL("&H"+A$)            C88
120 NEXT F                                    E52
130 SCREEN 0:LOCATE 0,7,0:KEY OFF             13E2
140 PRINT SPC(10);"[ 1 ] HEXAPRINTER"?"       1866
150 PRINT SPC(10);"[ 2 ] HEXASCREEN" "        23CA
160 A$=INKEY$                                 26D3
170 IF A$="1" THEN 260                        3002
180 IF A$<>"2" THEN 160                       3915
190    POKE &HD03B,&HA2                       3FA4
200    POKE &HD04A,&HA2                       46AF
210    POKE &HD04F,&HA2                       4E4C
220    POKE &HD058,&HC                        56E5
230    POKE &HD05C,&HA2                       5F9F
240    POKE &HD061,&HA2                       6835
250    POKE &HD070,&HC                        7173
260 DEFUSR=&HD000 : S=USR(0)                  79E5
270 SCREEN 0 : NEW                            7C1E
280 DATA F3,DD,21,B6,FF,DD,36,00              8100
290 DATA C3,DD,36,01,13,DD,36,02              87E8
300 DATA D0,FB,C9,E5,D5,C5,F5,FE              9069
310 DATA 0A,20,05,21,70,D0,36,01              9928
320 DATA 21,B6,FF,36,C9,47,3E,F0              A805
330 DATA A0,CB,3F,CB,3F,CB,3F,CB              BF5D
340 DATA 3F,C6,30,FE,3A,38,02,C6              D32C
350 DATA 07,C5,CD,A5,00,C1,3E,0F              E856
360 DATA A0,C6,30,FE,3A,38,02,C6              FA98
370 DATA 07,CD,A5,00,3E,20,CD,A5              F4
380 DATA 00,21,70,D0,35,20,0C,36              448
390 DATA 10,3E,0D,CD,A5,00,3E,0A              C90
400 DATA CD,A5,00,21,B6,FF,36,C3              1574
410 DATA F1,C1,D1,E1,33,33,B7,C9              2323
420 DATA 10,52,52,45,4E,41,54,4F              3407
```

5.5 - FILTRO GENÉRICO

O programa apresentado a seguir permite compatibilizar os caracteres acentuados de seu MSX (Expert 1.1 ou Hotbit) com a sua impressora, desde que ela os possua.

Digite e grave o programa a seguir e depois execute-o comandando:

```
RUN
```

Ligue a impressora e pressione a tecla RETURN.

Serão impressos os caracteres correspondentes aos códigos de 128 (&H80) a 255 (&HFF) e seus respectivos códigos em hexadecimal. Não se assuste se durante essa impressão sua impressora "agir" de forma estranha pois pode ser que alguns dos caracteres enviados para ela correspondam a alguns de seus controles, como avanço de linha, beep, etc.

Quando a impressão terminar veja o resultado e pressione RETURN novamente. Serão listadas na tela as linhas de 620 a 710 do programa para que você as altere conforme o resultado obtido na listagem da impressora.

Procure cada caractere das linhas DATA na lista impressa e substitua o código 20 (da listagem original) pelo código que foi impresso.

Feitas todas as alterações (não se esqueça de pressionar RETURN após cada uma delas!), comande:

```
GOTO 360
```

Com isso será gravado um arquivo de nome "FILTRO.BIN". Ele é seu programa "filtro" e para rodá-lo, basta comandar:

```
BLOAD"FILTRO.BIN",R
```

Utilize-o toda vez que você pretender imprimir um texto com acentuação.

```
100 POKE &HF417,1:SCREEN 0                    383
110 WIDTH 38:KEYOFF                           75E
120 LOCATE 0,10:PRINT"  PREPARE SUA";         EBF
130 PRINT" IMPRESSORA E QUANDO ES-   ";       1892
140 PRINT" TIVER PRONTO APERTE A TECLA"       2019
150 PRINT"  <RETURN>."                        2650
160 A$=INPUT$(1)                              2840
170 IF ASC(A$)<>13 THEN 160                   3550
```

```
180 FOR F= 128 TO 255
190 LPRINT CHR$(F);" = ";HEX$(F)
200 NEXT
210 CLS
220 LOCATE0,7:PRINT"  VEJA O QUE SAIU";
230 PRINT" NA IMPRESSORA E           ";
240 PRINT"  ALTERE AS LINHAS DATA."
250 PRINT:PRINT"  DEPOIS DE ALTERADAS";
260 PRINT", COMANDE :                ";
270 PRINT"GOTO 300":PRINT
280 PRINT" APERTE A TECLA <RETURN>";
290 PRINT" PARA COMEÇAR"
300 A$=INPUT$(1)
310 IF ASC(A$)<>13 THEN 300
320 CLS:PRINT "  NÃO SE ESQUEÇA DO";
330 PRINT " <RETURN> APóS AS    ALTER";
340 PRINT "AÇÕES E DE COMANDAR GOTO ";
350 PRINT "360.":PRINT:LIST 620-710:END
360 CLS
370 LOCATE 0,10:PRINT"PREPARE O DISCO";
380 PRINT" PARA GRAVAR O PROGRAMA"
390 PRINT"FILTRO E TECLE RETURN ";
400 PRINT" QUANDO PRONTO"
410 A$=INPUT$(1)
420 IF ASC(A$)<>13 THEN 410
430 RESTORE 620
440 FOR L=0 TO 55
450 READ Z$
460 Z$="&H"+RIGHT$(Z$,2)
470 POKE &HD026+L,VAL(Z$)
480 NEXT
490 RESTORE 720
500 FOR E=&HD000 TO &HD025
510 READ Z$
520 POKE E,VAL("&H"+Z$)
530 NEXT
540 BSAVE"FILTRO.BIN",&HD000,&HD05D
550 CLS
560 LOCATE0,8:PRINT"   O PROGRAMA";
570 PRINT" ESTÁ GRAVADO.":PRINT
580 PRINT"   PARA RODÁ-LO, COMANDE:"
590 PRINT:PRINT"   BLOAD";CHR$(34);
600 PRINT"FILTRO.BIN";CHR$(34);",R"
610 PRINT:PRINT:END
620 DATA Ç=20,ü=20,é=20,â=20,Á=20,à=20
630 DATA ¨=20,ç=20,ê=20,í=20,ó=20,ú=20
640 DATA Â=20,Ê=20,Ô=20,À=20,É=20,æ=20
650 DATA Æ=20,ô=20,ö=20,ò=20,û=20,ù=20
660 DATA ÿ=20,ö=20,Ü=20,¢=20,£=20,¥=20
```

```
670 DATA Ç=20,ƒ=20,á=20,í=20,ó=20,ú=20       1CDD
680 DATA ñ=20,Ñ=20,ª=20,º=20,¿=20,⌐=20       2DFB
690 DATA ¬=20,½=20,¼=20,¡=20,«=20,»=20       41FE
700 DATA Ã=20,ã=20,Ĩ=20,ĩ=20,Õ=20,õ=20       5AB6
710 DATA Ũ=20,ũ=20                           63BB
720 DATA 21,B6,FF,3E,C3,77,23,11             6999
730 DATA 13,D0,73,23,72,21,17,F4             6DFF
740 DATA 3E,FF,77,FE,80,D8,FE,B8             7555
750 DATA D0,E5,D5,21,A6,CF,16,00             7EB5
760 DATA 5F,19,7E,D1,E1,C9                   858E

                              TOTAL =        858E
```

Como exemplo, apresentamos a seguir as linhas DATA de 620 a 710 preenchidas para compatibilizar um Expert 1.1 com uma impressora Mônica El6030.

```
620 DATA Ç=A6,ü=D9,é=C8,â=C3,Ä=A2,à=C1
630 DATA  =BB,ç=C6,ê=C9,î=AC,ô=D1,û=B7
640 DATA Â=A3,Ê=A9,Ô=B2,À=A1,è=A8,æ=D5
650 DATA Æ=B5,ô=D2,ö=D4,ò=D0,ü=D8,ù=D6
660 DATA ÿ=DA,ö=D4,Ü=B9,¢=BF,£=BC,¥=BE
670 DATA Ç=20,ƒ=20,á=C2,í=CC,ó=D1,ú=D7
680 DATA ñ=20,Ñ=20,ª=DC,º=DD,¿=DE,⌐=20
690 DATA ¬=20,½=20,¼=20,¡=20,«=20,»=20
700 DATA Ã=A4,ã=C4,Ĩ=20,ĩ=20,Õ=D3,õ=D3
710 DATA Ũ=20,ũ=20
```

5.6 - IMPEDINDO O USO DA IMPRESSORA

Muitos programadores tentam proteger ao menos a originalidade de seus programas BASIC dos milhares de piratas amadores que farteiam por este país simplesmente desativando a listagem do mesmo na tela. Ledo engano o daqueles que pensam que isso é eficaz! Basta comandar LLIST e a listagem será enviada para a impressora. Se, entretanto, o programa ao ser carregado, desativar o uso da impressora o problema estará resolvido. Para isso, basta inserir o código &HC3 na posição de memória &HFFB6. Experimente rodar o programinha exemplo listado abaixo e depois tente enviá-lo para a impressora.

```
10 POKE &HFFB6,&HC3                              37A
20 REM Tente me listar numa impressora !         F21
30 PRINT "Comande LLIST !"                       1537
40 END                                           1685
```

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - página 149.

5.7 - CÓPIA GRÁFICA

O programa listado a seguir imprime uma cópia fiel da tela presente no vídeo do micro mediante o simples pressionamento da tecla ESC.

Digite-o e grave-o. Depois, execute-o. Com isso a rotina em Linguagem de Máquina estará pronta para ser usada na memória do micro.

```
100 FOR F=&HE000 TO &HE25F                  598
110  READ A$ : POKE F,VAL("&H"+A$)          CAF
120 NEXT F                                  E49
130 DEFUSR=&HE000 : S=USR(0)                1593
140 SCREEN 0 : NEW                          190A
150 DATA 3A,CC,FD,FE,C9,C0,21,12            2283
160 DATA E0,22,CD,FD,3E,CD,32,CC            2B01
170 DATA FD,C9,FE,3A,C0,F5,C5,D5            31F8
180 DATA E5,ED,73,59,E2,0E,00,3A            4E01
190 DATA AF,FC,B7,21,F0,00,11,28            64B8
200 DATA 06,28,06,21,00,01,11,20            7B14
210 DATA 08,3E,1B,CD,97,E0,3E,4B            8CCA
220 DATA CD,97,E0,7D,CD,97,E0,7C            93A7
230 DATA CD,97,E0,06,00,CD,A1,E0            993C
240 DATA D5,C5,21,5B,E2,42,11,08            A410
250 DATA 00,C5,E5,06,08,7E,FE,08            AD66
260 DATA 3F,CB,11,19,10,F7,79,CD            B82C
270 DATA 97,E0,E1,C1,23,10,EA,C1            C043
280 DATA D1,04,78,BB,20,D7,3E,0D            DEC7
290 DATA CD,97,E0,3E,1B,CD,97,E0            F422
300 DATA 3E,41,CD,97,E0,3E,08,CD            8B5
310 DATA 97,E0,3E,0A,CD,97,E0,3E            131D
320 DATA 1B,CD,97,E0,0C,79,FE,18            1B8D
330 DATA 20,8D,E1,D1,C1,F1,C9,CD            1EA1
340 DATA A5,00,D0,ED,7B,59,E2,18            279E
350 DATA F1,C5,D5,E5,FD,E5,21,5B            2FA4
360 DATA E2,3E,40,36,00,23,3D,20            40F7
370 DATA FA,3A,AF,FC,B7,F5,C5,C4            53C9
380 DATA 73,E1,C1,69,26,00,29,29            69F3
390 DATA 29,5D,54,29,29,F1,F5,20            7FDE
400 DATA 01,19,58,19,EB,D6,02,79            9122
410 DATA 01,00,00,2A,24,F9,E5,2A            974E
420 DATA 22,F9,38,19,20,0C,2A,CB            9C16
430 DATA F3,E3,2A,C7,F3,E6,18,47            A534
440 DATA 18,0B,2A,D5,F3,E3,2A,D1            AE35
450 DATA F3,07,E6,06,4F,19,CD,4A            BCE4
460 DATA 00,6F,26,00,29,29,29,09            C059
470 DATA EB,FD,E1,FD,19,2A,C9,F3            E03A
480 DATA 19,0F,0F,0F,E6,1F,4F,06            F58D
490 DATA 00,3A,E6,F3,57,E6,0F,5F            9CF
```

```
500 DATA F1,E5,3D,20,08,2A,BF,F3      12F0
510 DATA 09,CD,4A,00,57,21,5B,E2      1848
520 DATA 06,08,FD,E5,E3,CD,4A,00      1EF4
530 DATA 4F,E1,FD,23,3A,AF,FC,D6      274A
540 DATA 02,38,15,28,0C,51,0E,F0      303F
550 DATA 78,FE,05,28,08,FD,2B,18      4294
560 DATA 07,E3,CD,4A,00,57,23,E3      5384
570 DATA C5,06,08,CB,11,34,35,20      6885
580 DATA 0D,7A,30,04,0F,0F,0F,0F      7DE4
590 DATA E6,0F,20,01,7B,77,23,10      8E9A
600 DATA EA,C1,10,BE,E1,FD,E1,E1      9568
610 DATA D1,C1,C9,78,07,07,07,C6      9925
620 DATA 07,47,79,07,07,07,C6,07      A458
630 DATA 4F,AF,CD,87,00,57,CD,4A      AC66
640 DATA 00,FE,D0,C8,D5,C5,CD,99      B3AC
650 DATA E1,C1,F1,3C,FE,20,20,EA      C051
660 DATA C9,91,2F,FE,27,D0,4F,23      DEC6
670 DATA CD,4A,00,5F,78,93,5F,9F      F47B
680 DATA 57,23,CD,4A,00,47,23,CD      8FB
690 DATA 4A,00,CB,7F,28,05,21,20      101D
700 DATA 00,19,EB,14,15,C0,E6,0F      1679
710 DATA C8,57,3A,E0,F3,CB,4F,0F      1D0C
720 DATA 3E,08,30,01,87,28,05,CB      27AF
730 DATA 80,CB,88,87,6F,C6,06,B9      30E6
740 DATA D8,BB,D8,79,D6,07,4F,7D      4324
750 DATA 26,08,38,08,91,FE,09,38      52CD
760 DATA 02,3E,08,67,7B,D6,07,5F      6791
770 DATA 7D,2E,08,38,08,93,FE,09      7D74
780 DATA 38,02,3E,08,6F,FD,21,5B      8E16
790 DATA E2,D5,CB,79,20,48,E5,FD      941F
800 DATA E5,CB,7B,20,38,FD,7E,00      98C9
810 DATA B7,20,32,C5,D5,E5,3A,E0      A219
820 DATA F3,0F,30,04,CB,39,CB,3B      AA8F
830 DATA CB,5B,28,04,CB,9B,CB,E1      B984
840 DATA 68,26,00,44,29,29,29,09      CA58
850 DATA ED,4B,26,F9,09,CD,4A,00      D048
860 DATA 1C,07,1D,20,FC,30,03,FD      F346
870 DATA 72,00,E1,D1,C1,FD,23,1C      7BF
880 DATA 2D,20,BE,FD,E1,E1,11,08      1023
890 DATA 00,FD,19,D1,0C,25,20,A9      1514
900 DATA C9,3E,41,4C,45,50,48,C9      108F

                         TOTAL   =    108F
```

Agora, teste a rotina executando o programinha a seguir e pressionando a tecla ESC.

```
10 SCREEN 2                           100D
20 FOR F=80 TO 1 STEP -10             6F2
```

```
30  CIRCLE (128,80),80,15,,,80/F        CC7
40  CIRCLE (128,80),80,15,,,F/80        13A5
50 NEXT F                                15AC
60 LINE (128,160)-(128,0)                1F18
70 LINE (48,80)-(208,80)                 2630
80 GOTO 80                               2945

                           TOTAL   =     2945
```

O programa copia para a impressora todas a regiões da tela (qualquer uma das SCREEN's) cuja cor possua um código superior a 7, inclusive os SPRITES.

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 6.
Programação Avançada em MSX - capítulo 3.

5.8 - CARACTERES DEITADOS

O programa listado adiante gera uma rotina em L.M. que imprime uma string em modo gráfico, porém gira, previamente, seus caracteres de 90°.

Isso pode ser de grande utilidade para produzir mensagens que devem ser lidas "em pé", de cima para baixo.

A string a ser impressa deve ser passada como parâmetro na função USR.

```
10 '*********************          59A
20 '* BY THE DOCTOR LUZ *          DC8
30 '*********************          164E
40 FOR L=&HC000 TO &HC08A          1DA6
50 READ A$:POKE L,VAL("&H"+A$)     2657
60 NEXT                            2899
70 DEF USR=&HC000                  2CE1
80 PRINT"RODAR PROGRAMA ?"         3C3C
90 A$=INPUT$(1)                    4281
100 IF A$="S" THEN GOTO 200 ELSE END   4E56
200 PRINT"DIGITE A STRING A SER IMPRESSA6627
"
210 INPUT A$                       69FC
220 A$=USR(A$)                     6F82
230 GOTO 80                        7439
1000 DATA 2A,F8,F7,46,05,CD,3F,C0  8238
1010 DATA 23,7E,5F,23,7E,57,EB,E5  8808
1020 DATA E1,23,E5,7E,C5,21,BF,1B  8E48
1030 DATA 5F,16,00,CB,23,CB,12,CB  978C
1040 DATA 23,CB,12,CB,23,CB,12,AF  9F8D
1050 DATA ED,5A,EB,CD,61,C0,06,08  AFC2
1060 DATA 21,00,C2,7E,CD,A5,00,23  C177
1070 DATA 10,F9,C1,10,D3,E1,C9,3E  D329
1080 DATA 1B,CD,A5,00,3E,4B,CD,A5  E857
1090 DATA 00,58,16,00,CB,23,CB,12  F038
1100 DATA CB,23,CB,12,CB,23,CB,12  484
1110 DATA 7B,CD,A5,00,7A,CD,A5,00  A37
1120 DATA C9,06,08,C5,21,00,C2,06  1224
1130 DATA 08,1A,17,CB,16,23,10,FA  185D
1140 DATA C1,13,10,EF,C9,CD,ED,00  25C4
1150 DATA D1,E1,38,07,23,E7,20,F2  3834
1160 DATA C3,F0,00,21,30,71,C3,1A  47F9
1170 DATA 70,7D,C5                 4FF3
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - páginas 140 a 143.

5.9 - STRINGS EM MODO GRÁFICO

O programa a seguir é quase igual ao da dica 5.9. porém a impressão da string é feita sem que os caracteres seja girados.

Lembre-se que tanto esta dica quanto a anterior só funcionarão se a impressora conectada ao micro entrar em modo gráfico segundo o padrão EPSON e tiver 9 agulhas em sua cabeça de impressão.

```
10 '*********************                59A
20 '* BY THE DOCTOR LUZ *                1C0
30 '*********************                164E
40 FOR L=&HC000 TO &HC072                1CFE
50 READ A$:POKE L,VAL("&H"+A$)           25AF
60 NEXT                                  2761
70 DEF USR=&HC000                        2C39
80 PRINT"RODAR PROGRAMA ?"               3B94
90 A$=INPUT$(1)                          41D9
100 IF A$="S" THEN GOTO 200 ELSE END     4DAE
200 PRINT"DIGITE A STRING A SER IMPRESSA 657F
"
210 INPUT A$                             6954
220 A$=USR(A$)                           6EDA
230 GOTO 80                              7391
5000 DATA 2A,F8,F7,46,05,CD,3C,C0        81E6
5010 DATA 23,7E,5F,23,7E,57,68,26        8758
5020 DATA 00,ED,5A,E5,E1,2B,E5,7E        8CEF
5030 DATA C5,21,C6,1B,5F,16,00,CB        9728
5040 DATA 23,CB,12,CB,23,CB,12,CB        9FC4
5050 DATA 23,CB,12,AF,ED,5A,06,08        B060
5060 DATA 7E,CD,A5,00,2B,10,F9,C1        C1C8
5070 DATA 10,DA,E1,C9,3E,1B,CD,A5        D3C9
5080 DATA 00,3E,4B,CD,A5,00,58,16        E963
5090 DATA 00,CB,23,CB,12,CB,23,CB        F053
5100 DATA 12,CB,23,CB,12,7B,CD,A5        45A
5110 DATA 00,7A,CD,A5,00,C9,06,08        994
5120 DATA C5,21,00,C2,06,08,1A,17        1157
5130 DATA CB,16,23,10,FA,C1,13,10        1A81
5140 DATA EF,C9,10                       1EC9

                              TOTAL =    1EC9
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - página 143.

5.A - MÁQUINA DE ESCREVER

O pequeno programa listado a seguir, após ser executado. faz com que todos os caracteres digitados no teclado apareçam na tela e também na impressora. Desse modo pode-se usar a impressora quase como uma máquina de escrever.

```
100 KEY1,CHR$(13)+CHR$(10)                      671
110 SCREEN 0 : WIDTH 40                         9E2
120 PRINT "F1 avança 1 linha!"                  FE4
130 PRINT "HOME/CLS avança uma página!"         1A47
140 FOR F=&HE000 TO &HE00E                      215C
150   READ A$:POKE F,VAL("&H"+A$)               2894
160 NEXT F : DEFUSR0=&HE000                     334F
170 POKE 0,USR0(0) : END                        3AE8
180 DATA CD,9F,00,CD,A2,00,CD,4D                4C4F
190 DATA 01,CD,B7,00,30,F2,C9,00                6088
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - páginas 139 e 140.

5.8 - IMPRESSOR DE PROGRAMAS

O programa apresentado a seguir permite a impressão de listagens de programas sem que eles passem pelo "picote" do formulário contínuo. Ele só funcionará se o formulário for o de uso padrão, com 66 linhas de impressão, e se o entrelinhamento da impressora não tiver sido alterado.

Para usar o programa, digite-o e rode-o. Ele irá gerar e gravar o programa em Linguagem de Máquina, responsável pelo "salto" sobre o "picote". Antes de comandar LLIST para listar um programa de mais de 66 linhas, carregue a rotina em Linguagem de Máquina comandando:

```
BLOAD"IMPRE.BIN",R
```

Depois, pode-se usar o LLIST normalmente.

```
100 FOR F%=&HE000 TO &HE03A                 5A5
110   READ A$:POKE F%,VAL("&H"+A$)          CF9
120 NEXT F%:DEFUSR0=&HE000                  1286
130 POKE 0,USR0(0)                          16AB
140 BSAVE"IMPRE.BIN",&HE000,&HE03A          1DC8
150 END                                     1FCE
160 DATA F3,3E,C3,32,B6,FF,21,0E            288F
170 DATA E0,22,B7,FF,FB,C9,FE,0A            39A5
180 DATA C0,3A,3A,E0,3C,32,3A,E0            4B04
190 DATA FE,3F,3E,0A,C0,3E,C9,32            5E1F
200 DATA B6,FF,3E,0A,CD,A5,00,CD            7221
210 DATA A5,00,CD,A5,00,CD,A5,00            8618
220 DATA 3E,C3,32,B6,FF,AF,32,3A            8018
230 DATA E0,C9,00,00,00,00,00,00            9168

                                 TOTAL =    9168
```

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - página 149.

6 DICAS PARA O DRIVE

O disk drive é sem dúvida um dos mais importantes periféricos no sistema MSX. Dentre as máquinas de 8 bits, provavelmente a que consegue um melhor aproveitamento dos drives são as do padrão MSX. Vamos ver alguns recursos do drive a partir do BASIC e do MSXDOS.

Aconselhamos que a numeração dos programas seja seguida à risca, pois em alguns casos pode-se tirar proveito disso. A sequência de programas das dicas entre a 6.2 e a 6.6 pode ser emendada através de comandos "MERGE", constituindo-se num único programa utilitário de disco. Leia mais sobre isso nas explicações da dica 6.2 .

A bibliografia recomendada para todas estas dicas é a mesma:

Usando o Disk Drive no MSX.
Sistema de Disco para MSX e
Drives Leopard de 3 1/2"

6.1 - Personalizando disquetes 135
6.2 - Cabeçalhos de arquivos 136
6.3 - Características de discos 139
6.4 - Mapa de discos 141
6.5 - Leitor de setores 145
6.6 - Leitor de arquivos 147
6.7 - Programador de funções 149
6.8 - CLS em MSXDOS 151
6.9 - BEEP em MSXDOS 152
6.A - "SWAP" de arquivos em MSXDOS 152

6.1 - PERSONALIZANDO DISQUETES

O pequeno programa apresentado a seguir grava uma mensagem de até 512 bytes no último setor de um disco.

Você pode usá-lo para "marcar" seus discos com seu nome ou com qualquer outra mensagem.

O texto da mensagem deve ser inserido nas linhas DATA's ao fim do programa e deve terminar sempre com o caractere "0".

```
100 X$=DSKI$(0,0)                            362
110 EN=PEEK(&HF351)+256*PEEK(&HF352)         DB6
120 LS=PEEK(EN+19)+256*PEEK(EN+20)-1         1A57
130 READ X$                                  1DD3
140 IF X$="0" THEN 190                       24A3
150 FOR F=EN TO EN+LEN(X$)-1                 2D0F
160   POKE F,ASC(MID$(X$,F-EN+1,1))          3BB0
170 NEXT F : EN=F                            42ED
180 GOTO 130                                 45C4
190 DSKO$ 0,LS                               4AAD
200 END                                      4BE5
210 REM ------------------------             5C7E
220 REM   Texto a ser gravado                6E17
230 REM ------------------------             83D2
240 DATA Este disquete é de                  8DC8
250 DATA propriedade exclusiva               97C2
260 DATA de seu próprio dono !               9E69
270 DATA 0                                   A094

                          TOTAL   =          A094
```

BIBLIOGRAFIA RECOMENDADA:

Usando o Disk Drive no MSX - páginas 115 e 116.
Sistema de Disco para MSX - páginas 64 e 65.
Drives Leopard de 3 " - páginas 98 e 99.

6.2 - CABEÇALHOS DE ARQUIVOS

O programa apresentado a seguir lê o cabeçalho de programas gravados em disco no formato binário. Após digitá-lo e conferí-lo, grave-o no formato ASCII.

```
2000 REM                                          162
2010 REM   cabeçalho de arquivos                  C10
2020 REM -------------------------------          FAA
2030 PLAY "S0M50000iC#"                           1505
2040 SCREEN 0 : WIDTH 38 : KEY OFF                1E93
2050 COLOR 15,1,1 : PRINT : PRINT                 2671
2060 PRINTSPC(9);"CABEÇALHO DE ARQUIVOS"          391A
2070 PRINTSPC(9);"---------------------"          49A6
2080 PRINT:FILES:PRINT:PRINT:X$=""                5085
2090 PRINT"NOME DO ARQUIVO ";:INPUT X$            6320
2100 IF X$="" THEN 1000                           6CA7
2110 OPEN X$ AS #1 LEN=1                          7806
2120 FIELD #1,1 AS A$                             7058
2130 IF LOF(1)<>0 THEN 2140                       83E0
2131   CLOSE #1 : KILL X$ : GOTO 2000             8940
2140 GET #1,1 : IF ASC(A$)=254 THEN 2160          9284
2150 PRINT"NÃO É BINÁRIO":GOTO 2260               98FE
2160 RESTORE 2250 : PRINT                         A0EA
2170 FOR F= 1 TO 6 STEP 2                         A7F4
2180 READ B$ : PRINT "--> ";B$;":";               B8F6
2190 GET #1,F+2                                   BC48
2200 PRINT "&H";                                  C10A
2210 PRINT RIGHT$("0"+HEX$(ASC(A$)),2);           CEAC
2220 GET #1,F+1                                   D591
2230 PRINT RIGHT$("0"+HEX$(ASC(A$)),2)            E4A5
2240 NEXT F : CLOSE #1                            ECC9
2250 DATA INÍCIO  ,FIM     ,EXECUÇÃO              FA0E

                                   TOTAL  =       FA0E
```

Para usar o programa acima e os programas das dicas de 6.3 a 6.6 será necessário digitar também o programa MENU, listado a seguir. Digite-o e grave-o em ASCII com o nome MENU.

```
240 FOR F=1 TO 10 : KEY F,"" : NEXT F             B6E
1000 REM                                          A11
1010 REM MENU PRINCIPAL                           1178
1020 REM -------------------------                1AAA
1030 PLAY "S8M40005G#"                            202F
1040 COLOR 15,1 : SCREEN 0 : WIDTH 40             2903
1050 KEY OFF : CLOSE                              2049
1060 FOR F=1 TO 8: KEY (F) ON : NEXT F            3886
```

```
1070 LOCATE 9,2,0                                       40FE
1080 PRINT ">>> MENU PRINCIPAL <<<"                     5568
1090 PRINT SPC(9);STRING$(22,"-")                       6240
1100 PRINT,,,,                                          67C0
1110 PRINT SPC(7);" [ F1 ] MENU PRINCIPA                 808C
L": PRINT
1120 PRINT SPC(7);" [ F2 ] CABEÇALHO DE                 8F78
ARQUIVOS": PRINT
1130 PRINT SPC(7);" [ F3 ] CARACTERíSTIC                98A5
AS": PRINT
1140 PRINT SPC(7);" [ F4 ] MAPA DO DISQU                A68C
ETE": PRINT
1150 PRINT SPC(7);" [ F5 ] DUMP DE SETOR                B7FA
ES": PRINT
1160 PRINT SPC(7);" [ F6 ] DUMP DE ARQUI                D22B
VOS": PRINT
1170 PRINT SPC(7);" [ F7 ] EDITAR DISCO                 E787
": PRINT
1180 PRINT SPC(7);" [ F8 ] TERMINAR": PR                F8FA
INT
1190 ON KEY GOSUB 1000,2000,3000,4000,50                A18
00,6000,7000,8000
1200 GOTO 1200                                          C96
2260 PRINT : LOCATE 8,23,0                              106C
2270 PRINT">>> TECLE ESPAÇO <<<";                       1980
2280 IF STRIG(0)=0 THEN 2280 ELSE RETURN                274C
 1000
3460 PRINT">>> TECLE ESPAÇO <<<"                        2F80
3470 IF STRIG(0)=0 THEN 3470 ELSE RETURN                305C
 1000
4920 LOCATE8,23,0                                       4263
4930 PRINT ">>> TECLE ESPAÇO <<<";                      5408
4940 IF STRIG(0)=0 THEN 4940 ELSE RETURN                65AF
 1000
5500 LOCATE8,22,0                                       6C40
5510 PRINT">>> TECLE ESPAÇO <<<";                       7E02
5520 IF STRIG(0)=0 THEN 5520 ELSE GOTO 1                8A10
000
6430 LOCATE8,22,0                                       889E
6440 PRINT ">>> TECLE ESPAÇO <<<"                       926C
6450 IF STRIG(0)=0 THEN 6450 ELSE GOTO 1                9CE9
000
7000 REM                                                9E32
7010 REM          Editar disco                          A487
7020 REM ------------------------------                 B2AC
7030 SCREEN 0                                           B502
7040 PRINT : PRINT : PRINT SPC(10);                     BE28
7050 PRINT "NÃO IMPLEMENTADA !!!"                       D080
7060 PRINT : PRINT : PRINT SPC(10);                     DA00
```

```
7070 PRINT "  DIGITE RETURN !!!"             E922
7080 COLOR (F) MOD 15 : F=F+1                F904
7090 IF INKEY$=CHR$(13) THEN RUN             FE08
7100 BEEP : GOTO 7080                        26E
8000 REM                                     367
8010 REM   terminar                          830
8020 REM ------------------------------      CE6
8030 SCREEN 0 : WIDTH 40 : KEY ON            1502
8040 DEFUSR0=&H3E : X=USR0(0)                1CFD
8050 FOR F=1 TO 8 : KEY(F) OFF : NEXT F      278F
8060 END                                     292D

                             TOTAL  =        292D
```

Quando você já tiver todos os programas (6.2, 6.3, 6.4, 6.5 e 6.6) digitados e gravados, digite a seguinte sequência de comandos:

```
LOAD "MENU"
MERGE "programa 6.2"
MERGE "programa 6.3"
MERGE "programa 6.4"
MERGE "programa 6.5"
MERGE "programa 6.6"
```

Depois, grave o programa presente na memória do micro com o nome "DISCOUTI.BAS". Rode-o para ver como as dicas operam em conjunto.

A "SOMA TOTAL" do programa completo deverá ser:

6.3 - CARACTERÍSTICAS DE DISCOS

A rotina apresentada adiante lê o setor 0 de um disco e fornece as características de formatação do mesmo. Após digitar e conferir a listagem, grave-a no formato ASCII para posterior utilização com o comando MERGE.

```
3000 REM                                          14E
3010 REM    Características do disco             D1F
3020 REM ---------------------------------        1241
3030 PLAY "S0M50000 2D#"                          1884
3040 SCREEN 0 : WIDTH 38 : KEY OFF                20A6
3050 PRINT"CARACTERÍSTICAS DO DISQUETE"           2062
3060 PRINT STRING$(27,"-") : PRINT                36D7
3070 PRINT "Fornecedor O&M ..... :"               4940
3080 PRINT "Bytes por setor .... :"               5E9A
3090 PRINT "Setores por bloco .. :"               7236
3100 PRINT "Setores reservados . :"               810E
3110 PRINT "Nº de F.A.T.'s ..... :"               87CE
3120 PRINT "Entr. no diretório . :"               8E79
3130 PRINT "Setores no disco ... :"               9756
3140 PRINT "Tipo de disco ...... :"               9DCC
3150 PRINT "Setores por F.A.T. . :"               B016
3160 PRINT "Setores por trilha . :"               C465
3170 PRINT "Faces .............. :"               D9FE
3180 PRINT "Setores ocultos .... :"               EE58
3190 A$=DSKI$(0,0) : C = 256                      F08D
3200 E=C*PEEK(&HF352)+PEEK(&HF351)                938
3210 DEFFNA(X)=C*PEEK(E+X)+PEEK(E+X-1)            171B
3220 DEFFNB(X)=PEEK(E+X) : LOCATE 23,3            219B
3230 FOR I=E+3 TO E+10                            288F
3240      PRINT CHR$(PEEK(I));                    31EE
3250 NEXT I                                       3487
3260 LOCATE 22,4 : PRINT FNA(12)                  3D50
3270 LOCATE 22,5 : PRINT FNB(13)                  4821
3280 LOCATE 22,6 : PRINT FNA(15)                  52EB
3290 LOCATE 22,7 : PRINT FNB(16)                  5DEB
3300 LOCATE 22,8 : PRINT FNA(18)                  681E
3310 LOCATE 22,9 : PRINT FNA(20)                  7887
3320 LOCATE 23,10: PRINT HEX$(FNB(21))            88AE
3330 LOCATE 22,11: PRINT FNA(23)                  92C8
3340 LOCATE 22,12: PRINT FNA(25)                  9720
3350 LOCATE 22,13: PRINT FNA(27)                  986D
3360 LOCATE 22,14: PRINT FNA(29)                  A10E
3370 PRINT ,,"Capacidade total ... :"             AA4A
3380 BS = FNA(12) : SE = FNA(20)                  B396
3390 LOCATE 22,16,0                               B981
3400 PRINT USING "#########";SE*BS;               C95C
```

```
3410 PRINT " bytes"                                  D135
3420 PRINT "Área ainda disponível:"                   E545
3430 D=DSKF(0):S=FNB(13):LOCATE 22,17                 F6C3
3440 PRINT USING "#########";D*S*BS;                  7538
3450 PRINT " bytes":PRINT:LOCATE 8,21,0               109C

                                          TOTAL   =   109C
```

```
CARACTERÍSTICAS DO DISQUETE
---------------------------

Fornecedor O&M ..... : MSX-02
Bytes por setor .... : 512
Setores por bloco .. : 2
Setores reservados . : 1
Nº de F.A.T.'s ..... : 2
Entr. no diretório . : 112
Setores no disco ... : 720
Tipo de disco ...... : FD
Setores por F.A.T. . : 2
Setores por trilha . : 9
Faces .............. : 2
Setores ocultos .... : 0

Capacidade total ... :   368640 bytes
Área ainda disponível:   205824 bytes


Ok
█
```

6.4 - MAPA DE DISCOS

O programa apresentado a seguir lê a FAT do disco presente no drive A e apresenta na tela o conteúdo de cada uma de suas posições. Em seguida o diretório do disco é lido e cada um de seus arquivos são mostrados na tela com todos os seus parâmetros especificados, incluindo os blocos do disco em que ele está armazenado.

Após digitar e conferir o programa, grave-o no formato ASCII para posterior utilização com o comando MERGE.

Caso um arquivo tenha sido apagado com o comando KILL do DISK BASIC ou ERASE ou DEL do MSXDOS, o programa permitirá a sua recuperação, mas APENAS NO DIRETÓRIO. Se o arquivo apagado ocupar menos que um bloco do disco, ele será automaticamente recuperado também na FAT, mas se ele ocupar mais de um bloco, apesar de ser recuperado no diretório, não será mapeado por completo na FAT.

```
4000 REM                                      13A
4010 REM mapa do disquete                     8CC
4020 REM --------------------------------     C8D
4030 PLAY "S0M500003E#"                       1228
4040 SCREEN 0 : WIDTH 36 : KEY OFF            1AD8
4050 PRINT SPC(6);">>> MAPA DO DISQUETE       2506
<<<"
4060 PRINT SPC(6);STRING$(24,"-")             2E98
4070 PRINT:PRINT:PRINT                        30FC
4080 PRINT SPC(6);" >>> LENDO A F.A.T. <      4A74
<<":LOCATE0,3,0
4090 GOSUB 8260                               4ECD
4100 FOR F=5 TO 11                            56AB
4110    LOCATE0,2,0:PRINTCHR$(27)"J"          6606
4120    PRINTSPC(9);">>> SETOR";              75CA
4130    PRINT USING "###";F;                  7C4A
4140    PRINT " <<<"                          7ECF
4150    PRINTSPC(9);"=================="      8297
4160    A$ = DSKI$(0,F)                       86EC
4170    P = PEEK(&HF351)+256*PEEK(&HF352)     92AD
4180    FOR G=0 TO 15                         9860
4190    LOCATE0,5,0:PRINTCHR$(27)"J";         A1A9
4200    IF INKEY$=CHR$(27) THEN 1000          AC40
4210       PRINT "ARQUIVO";(F-5)*16+G+1       C052
4220       PRINT "-----------"                CE8A
4230 REM ----------------------------         E8E6
4240       PRINT"NOME          : ";           FA55
4250       C$ = ""                            FF91
```

```
4260      FOR H=0 TO 10                         47F
4270        A = P + 32*G + H                    BE2
4280        IF PEEK (A)<>0 THEN 4310            14BB
4290        PRINTSTRING$(95,127)                1BF2
4300         GOTO 4920                          21C9
4310        C$ = C$ + CHR$(PEEK(A))             2E49
4320        IF H=7 THEN C$=C$+"."               3067
4330      NEXT H                                423F
4340      PRINT C$                              4780
4350 REM ---------------------------            5024
4360      PRINT"ATRIBUTOS  :";                  6CFD
4370      AT = PEEK ( P + 32*G + 11 )           7FF9
4380      AT$ = RIGHT$("00000000"+BIN$(AT       89F2
),8)
4390      PRINT " ";AT$                         8C39
4400 REM ---------------------------            95AC
4410      PRINT"HORA          :";               9CCA
4420      X=(P+32*G+22)                         A389
4430      X1=PEEK(X)                            AAF6
4440      X2=PEEK(X+1)                          B49A
4450      X1$=RIGHT$("00000000"+BIN$(X1),       C79F
8)
4460      X2$=RIGHT$("00000000"+BIN$(X2),       E029
8)
4470      H$=LEFT$(X2$,5)                       ECC7
4480      M$=RIGHT$(X2$,3)+LEFT$(X1$,3)         FCF9
4490      H=VAL("&B"+H$)                        3F3
4500      M=VAL("&B"+M$)                        978
4510      PRINT " ";H;M                         C8A
4520 REM ---------------------------            19F5
4530      PRINT"DATA          :";               214B
4540      H2=PEEK(P + 32*G + 24)                2E77
4550      H1=PEEK (P + 32*G + 25)               3E76
4560      H1$=RIGHT$("00000000"+BIN$(H1),       51CB
8)
4570      H2$=RIGHT$("00000000"+BIN$(H2),       6A11
8)
4580      D=VAL("&B"+RIGHT$(H2$,5))             77F5
4590      M=VAL("&B"+RIGHT$(H1$,1)+LEFT$(       8478
H2$,3))
4600      A=1980+VAL("&B"+LEFT$(H1$,7))         9028
4610      PRINT D;M;A                           94CC
4620 REM ---------------------------            9FAC
4630      PRINT"1º BLOCO    :  ";               AB77
4640      H=(P+32*G+26)                         B743
4650      H=PEEK(H)+256*PEEK(H+1)               C526
4660      PB=H                                  CB0E
4670      PRINT RIGHT$("000"+HEX$(H),3)         DE22
4680 REM ---------------------------            F632
```

```
4690      PRINT"Nº DE BYTES: ";                        92
4700      H=(P+32*G+28)                                 74C
4710      H=PEEK(H)+256*PEEK(H+1)+                      1FD7
        4096*PEEK(H+2)+65536!*PEEK(H+3)
4720      PRINT RIGHT$("0000"+HEX$(H),4)                2A59
4730 REM ---------------------------------              3002
4740      PRINT"MAPA DA FAT: ";                         40B2
4750      H=PB                                          541A
4760      IF H>359 THEN 4830                            5F61
4770      PRINT RIGHT$("000"+HEX$(B%(H)),               78Y1
3);" ";
4780      IF H>359 THEN 4830                            8475
4790      H=B%(H)                                       8803
4800      IF H>359 THEN 4830                            8EFB
4810      PRINT RIGHT$("000"+HEX$(B%(H)),               988F
3);" ";
4820      GOTO 4780                                     90F6
4830 REM ---------------------------------              A98C
4840      PRINT                                         AC8A
4850      PRINT                                         AFF1
4860      IF LEFT$(C$,1)=CHR$(&HE5) THEN                BF88
GOSUB 8070
4870    LOCATE10,23,0                                   C757
4880    PRINT"DIGITE RETURN:";                          D807
4890    A$=INPUT$(1)                                    E078
4900    NEXT G                                          E5EA
4910 NEXT F                                             E981
4920 LOCATE8,23,0                                       F048
4930 PRINT ">>> TECLE ESPAÇO <<<";                      400
4940 IF STRIG(0)=0 THEN 4940 ELSE RUN                   BFF
8070 REM                                                D40
8080 REM Recupera deletados                             14EF
8090 REM ---------------------------------              1C29
8100 PRINT,,,"> Arquivo deletado ! <"                   24EA
8110 PRINT,"RECUPERAR (S/N)?:";                         2C2C
8120 BEEP : BEEP : BEEP                                 31DC
8130 X = PEEK(&HFCAB) : POKE &HFCAB,1                   3083
8140 X$ = INPUT$(1) : PRINT X$                          480F
8150 IF X$<>"S" THEN 8220                               5298
8160    PRINT,"1º caractere do nome:";                  6885
8170    Z$ = INPUT$(1) : PRINT Z$                       7718
8180    IF Z$<"A" OR Z$>"Z" THEN 8160                   82F5
8190    POKE (A-10),ASC(Z$)                             891F
8200    DSKO$ 0,F                                       8816
8210    GOTO 8230                                       8CA1
8220 IF X$<>"N" THEN 8110                               9366
8230 POKE &HFCAB,X : PRINT : PRINT                      9C11
8240 RETURN                                             9DCB
8250 REM                                                9F9D
```

```
8260 REM leitor de F.A.T.                          A3C5
8270 REM ------------------------------            B6CC
8280 A%(0)=B%(0):ERASE A%,B%                       C428
8290 DIM A%(539),B%(359)                           D08A
8300 A$=DSKI$(0,1)                                 D817
8310 P = PEEK(&HF351)+256*PEEK(&HF352)             E6B3
8320 FOR F=0 TO 511                                EE5C
8330   A%(F) = PEEK(P+F)                           F958
8340 NEXT F                                        FAFD
8350 A$=DSKI$(1,2)                                 FE9D
8360 P = PEEK(&HF351)+256*PEEK(&HF352)             9BE
8370 FOR F=0 TO 27                                 DFF
8380   A%(F+512) = PEEK(P+F)                       16B8
8390 NEXT F                                        1907
8400 G=0                                           1B0F
8410 FOR F=0 TO 539 STEP 3                         230F
8420  IF INKEY$=CHR$(27) THEN 1000                 2C5A
8430  B%(G)  =A%(F)+256*(A%(F+1) AND &HF           4226
)
8440  B%(G+1)=(A%(F+1) AND &HF0)/16 +              6A26
               A%(F+2)*16
8450  G=G+2                                        6F6A
8460 NEXT F                                        72EB
8470 FOR F=1 TO 360                                7BD3
8480  IF INKEY$=CHR$(27) THEN 1000                 845F
8490   PRINT RIGHT$("000"+HEX$(F-1),3);"           96BC
W";RIGHT$("000"+HEX$(B%(F-1)),3);"  ";
8500   IF F/72 <> F\72 THEN 8550                   A0B8
8510    LOCATE 10,23,1                             A6BC
8520    PRINT "DIGITE RETURN:";                    ADCD
8530    A$=INPUT$(1)                               B5AE
8540    LOCATE 0,3,0                               BD57
8550 NEXT F                                        C0C6
8560 PRINT CHR$(27);"J"                            C852
8570 RETURN                                        C9F2

                              TOTAL   =            C9F2
```

```
>>> MAPA DO DISQUETE <<<
------------------------

>>> SETOR  5 <<<
================
ARQUIVO 6
----------
NOME          : FILTRO2 .ASM
ATRIBUTOS     : 00000000
HORA          :  0  0
DATA          : 25  4  1986
1º BLOCO      :  009
Nº DE BYTES   : 0A80
MAPA DA FAT   : 00A 00B FFF
```

6.5 - LEITOR DE SETORES

O programa a seguir gera na tela um "DUMP" do setor de disco especificado pelo usuário. Após digitá-lo e conferí-lo, grave-o em formato ASCII para posterior utilização com o comando MERGE.

```
100 CLEAR 1000,&HC000                              325
105 ONERROR GOTO 100                               677
110 RESTORE 150                                    890
120 FOR F%=&HC000 TO &HC02D                        E36
130 READ X$ : POKE F%,VAL("&H"+X$)                 15B0
140 NEXT F% : DEFUSR0=&HC000                       106A
150 DATA 21,00,00,11,2E,C0,01,C0                   2730
160 DATA 03,CD,59,00,01,58,02,21                   3551
170 DATA F6,C0,11,CE,C0,ED,B0,06                   473F
180 DATA 28,21,4E,C3,36,20,23,10                   5742
190 DATA FB,21,2E,C0,11,00,00,01                   605F
200 DATA C0,03,CD,5C,00,C9,F3,21                   8136
210 DATA 52,45,4E,41,54,4F,20,44                   8ACC
220 DATA 41,20,53,49,4C,56,41,20                   8F65
230 DATA 4F,4C,49,56,45,49,52,41                   96AD
5000 REM                                           9888
5010 REM   dump de setores                         9F28
5020 REM ------------------------------            A893
5030 PLAY"S0M50000SG#"                             B61C
5040 SCREEN 0 : WIDTH 39                           BC9F
5050 X$=DSKI$(0,0)                                 C263
5060 E=PEEK(&HF351)+256*PEEK(&HF352)               CE70
5070 NS=256*PEEK(E+20)+PEEK(E+20-1)-1              DED1
5080 S$=""                                         E2AF
5090 PRINT "ENTRE O SETOR : &H  ";                 F638
5100 PRINT CHR$(8);CHR$(8);                        FC3E
5110 X$=INPUT$(1) : PRINT X$;                      1A3
5120 IF X$=CHR$(8) THEN 5000                       7F2
5130 S$=S$+X$                                      B6F
5140 IF LEN(S$)<3 THEN 5110                        13A3
5150 S$=RIGHT$("00"+S$,3)                          1890
5160 S=VAL("&H"+S$)                                2163
5170 IF S<0 OR S>NS THEN 5000                      294C
5180 CLS                                           2AAF
5190 PRINT : PRINT " SETOR : &H";S$                3AF6
5200 PRINT : PRINT STRING$(39,"-")                 43E7
5210 LOCATE 0,21:PRINT STRING$(39,"-")             512C
5220 X$=DSKI$(0,S)                                 5894
5230 EN=PEEK(&HF351)+256*PEEK(&HF352)              6604
5240 FOR F=0 TO 63                                 6E01
5250   IF INKEY$=CHR$(27) THEN 1000                770E
5260   IF F/16<>F\16 THEN 5320                     7F7C
```

```
5270       LOCATE 10,23,0                            8217
5280       PRINT "DIGITE RETURN:";                   898A
5290       Y$=INPUT$(1)                              902F
5300       LOCATE 10,23,0                            96CE
5310       PRINT "                    ";             A09C
5320     X=USR(0)                                    A68F
5330     LOCATE 0,19,0                               A07F
5340     PRINT RIGHT$("00"+HEX$(8*F),3);             BA87
5350     FOR G=0 TO 7                                C1A0
5360       X=PEEK(EN+8*F+G)                          CB09
5370       Y$=RIGHT$("0"+HEX$(X),2)                  DC3F
5380       PRINT " ";Y$;                             E7E8
5390     NEXT G                                      ED88
5400     PRINT " : ";                                F303
5410     FOR G=0 TO 7                                F6BE
5420       X=PEEK(EN+8*F+G)                          FF07
5430       Y$=CHR$(X)                                319
5440       IF X>31 THEN 5460                         809
5450         Y$=CHR$(1)+CHR$(64+X)                   1304
5460       PRINT Y$;                                 17B1
5470     NEXT G                                      1A76
5480     PRINT                                       1C88
5490 NEXT F                                          1E91

                                  TOTAL   =          1E91
```

```
 SETOR : &H005

---------------------------------------------
080 43 45 4D 31 2D 33 20 20 : CEM1-3
088 42 41 53 00 00 00 00 00 : BAS
090 00 00 00 00 00 00 00 00 :
098 99 0C 06 00 80 00 00 00 : ö♀♠ Ç
0A0 46 49 4C 54 52 4F 32 20 : FILTRO2
0A8 41 53 4D 00 00 00 00 00 : ASM
0B0 00 00 00 00 00 00 00 00 :
0B8 99 0C 09 00 80 0A 00 00 : ö♀○ Ç◙
0C0 46 49 4C 54 52 4F 32 20 : FILTRO2
0C8 42 49 4E 00 00 00 00 00 : BIN
0D0 00 00 00 00 00 00 00 00 :
0D8 99 0C 0C 00 F3 00 00 00 : ö♀♀ ≤
0E0 43 45 4D 31 2D 34 20 20 : CEM1-4
0E8 42 41 53 00 00 00 00 00 : BAS
0F0 00 00 00 00 00 00 00 00 :
0F8 99 0C 07 00 80 01 00 00 : ö♀• Ç☺
---------------------------------------------

           DIGITE RETURN:█
```

6.6 - LEITOR DE ARQUIVOS

O programa a seguir gera na tela um "DUMP" do arquivo especificado pelo usuário.

Note que as linhas entre 100 e 230 são idênticas às do programa leitor de setores.

Após digitar e conferir o programa, grave-o em formato ASCII para posterior utilização com o comando MERGE.

```
100 CLEAR 1000,&HC000                            325
105 ONERROR GOTO 100                             677
110 RESTORE 150                                  890
120 FOR F%=&HC000 TO &HC02D                      E36
130 READ X$ : POKE F%,VAL("&H"+X$)               15B0
140 NEXT F% : DEFUSR0=&HC000                     106A
150 DATA 21,00,00,11,2E,C0,01,C0                 2730
160 DATA 03,CD,59,00,01,58,02,21                 3551
170 DATA F6,C0,11,CE,C0,ED,B0,06                 473F
180 DATA 28,21,4E,C3,36,20,23,10                 5742
190 DATA FB,21,2E,C0,11,00,00,01                 6D5F
200 DATA C0,03,CD,5C,00,C9,F3,21                 8136
210 DATA 52,45,4E,41,54,4F,20,44                 8ACC
220 DATA 41,20,53,49,4C,56,41,20                 8F65
230 DATA 4F,4C,49,56,45,49,52,41                 96AD
6000 REM                                         9807
6010 REM   dump de arquivos                      90FB
6020 REM ------------------------------          AAB6
6030 PLAY"S0M500003D#"                           B534
6040 SCREEN 0 : WIDTH 39                         BBA5
6050 PRINT" ARQUIVOS:" : PRINT : FILES           C9F7
6060 PRINT : PRINT                               CCBB
6070 PRINT "Entre o nome do arquivo:";           E25A
6080 INPUT X$                                    E617
6090 IF X$="" THEN 1000                          EF2A
6100 CLS                                         F0A0
6110 PRINT : PRINT "ARQUIVO :";X$                F83C
6120 PRINT : PRINT STRING$(39,"-")               A4
6130 LOCATE 0,21:PRINT STRING$(39,"-")           51C
6140 OPEN X$ AS #1 LEN=1                         AF9
6150 FIELD #1,1 AS Y$                            FDD
6160 LF=INT(LOF(1)/8)+1                          1881
6170 FOR F=0 TO LF                               107E
6180    IF INKEY$=CHR$(27) THEN 1000             2731
6190    IF F/16<>F\16 THEN 6250                  3303
6200       LOCATE 10,23,0                        3050
6210       PRINT "DIGITE RETURN:";               4FC9
6220       X$=INPUT$(1)                          5A57
6230       LOCATE 10,23,0                        669C
```

```
6240      PRINT "                      ";    7C24
6250    X=USR0(0)                          8310
6260    LOCATE 0,19,0                      8586
6270    PRINT RIGHT$("000"+HEX$(1+8*F),4) 8F02
;
6280    X$=""                              9150
6290    FOR G=F*8+1 TO F*8+8               9A02
6300      IF G<=LOF(1) THEN 6340           A507
6310        Z$="  "                        AA40
6320        X$=X$+" "                      B0AE
6330        GOTO 6390                      B7AE
6340      GET #1,G                         BED4
6350      X=ASC(Y$)                        C6D2
6360      IF X>31 THEN X$=X$+CHR$(X)       D5D6
6370      IF X<32 THEN X$=X$+CHR$(1)+CHR$ EA18
(64+X)
6380      Z$=RIGHT$("0"+HEX$(X),2)         F8E6
6390      PRINT " ";Z$;                    FF39
6400    NEXT G                             A3
6410    PRINT "| ";X$                      3B7
6420 NEXT F                                4D5

                           TOTAL   =       4D5
```

```
ARQUIVO :CEM6-A.BAS

---------------------------------------------
0001 31 30 30 20 43 4C 45 41 | 100 CLEA
0009 52 20 31 30 30 30 2C 26 | R 1000,&
0011 48 43 30 30 30 0D 0A 31 | HC000♪◙1
0019 30 35 20 4F 4E 45 52 52 | 05 ONERR
0021 4F 52 20 47 4F 54 4F 20 | OR GOTO
0029 31 30 30 0D 0A 31 31 30 | 100♪◙110
0031 20 52 45 53 54 4F 52 45 |  RESTORE
0039 20 31 35 30 0D 0A 31 32 |  150♪◙12
0041 30 20 46 4F 52 20 46 25 | 0 FOR F%
0049 3D 26 48 43 30 30 30 20 | =&HC000
0051 54 4F 20 26 48 43 30 32 | TO &HC02
0059 44 0D 0A 31 33 30 20 52 | D♪◙130 R
0061 45 41 44 20 58 24 20 3A | EAD X$ :
0069 20 50 4F 4B 45 20 46 25 |  POKE F%
0071 2C 56 41 4C 28 22 26 48 | ,VAL("&H
0079 22 2B 58 24 29 0D 0A 31 | "+X$)♪◙1

---------------------------------------------

           DIGITE RETURN:
```

6.7 - PROGRAMADOR DE FUNÇõES

O programa apresentado a seguir permite a programação das teclas de função de forma bastante simples. Após digitá-lo e conferí-lo, grave-o com o nome "FUNKEY.BAS".

O uso deste programa é mais indicado para o MSXDOS. Nesse caso, deve-se alterar o "END" da linha 640 para "_SYSTEM" e gerar um arquivo BAT com o comando "BASIC FUNKEY.BAS". Assim, estando em DOS, bastará executar o arquivo BAT para poder programar as teclas de função.

```
100 CLEAR 500 : DEFINT A-Z                                  522
110 SCREEN 0,,0 : WIDTH 38 : KEY OFF                        B29
120 FOR F=1 TO 4 : KEY(F) ON : NEXT F                       1394
130 ON KEY GOSUB 490,650,760,790                            1D8C
140 PRINT"REDEFINIDOR DE FUNCOES";                          241B
150 PRINT" - EDITORA ALEPH";                                2F1B
160 PRINTSTRING$(38,"");                                    3510
170 PRINT : PRINT : PRINT : PRINT                           3C05
180 PRINT CHR$(27);"y4"                                     443F
190 FOR F=1 TO 10                                           4ABE
200   PRINT SPC(7);                                         51C7
210   PRINT "F";USING"##";F;                                5F9C
220   PRINT " --->[                    ]"                   76A5
230 NEXTF                                                   78F0
240 PRINT : PRINT                                           7B78
250 PRINT SPC(11); "F1 - VOLTA AO DOS"                      8384
260 PRINT SPC(11); "F2 - REINICIALIZA"                      8C3B
270 PRINT SPC(11); "F3 - DEFAULT BASIC"                     943A
280 PRINT SPC(11); "F4 - LIMPA TEXTOS"                      9C10
290 X=16 : Y=7                                              A209
300 LOCATE X,Y,1                                            A898
310 A$=INKEY$                                               AC60
320 IF A$<CHR$(28) THEN 350                                 B724
330 IF A$>CHR$(31) THEN 350                                 C232
340 A$=""                                                   C740
350 A=STICK(0)                                              CC95
360 IF A>0 AND A<3 OR A=8 THEN Y=Y-1                        DC66
370 IF Y=6 THEN Y=16                                        E54F
380 IF A>3 AND A<7 THEN Y=Y+1                               F11E
390 IF Y=17 THEN Y=7                                        F695
400 IF A>1 AND A<5 THEN X=X+1                               FF91
410 IF X>30 THEN X=16                                       59E
420 IF A>5 THEN X=X-1                                       C3F
430 IF X=15 THEN X=30                                       12C6
440 IF A$=""THEN 300                                        1966
450 LOCATE,,0                                               1C25
```

```
460 VPOKE 40*Y+X+1,ASC(A$)                          2452
470 IF X<30 THEN X=X+1                              2D5D
480 GOTO 300                                        310D
490 BEEP : BEEP : BEEP                              376A
500 FOR F=1 TO 10                                   3D84
510   LOCATE 0,0,0                                  43EE
520   A$="" : FL=0                                  4EBA
530   X=(F+6)*40+17                                 59FB
540   FOR Y=X+14 TO X STEP -1                       6785
550     IF FL=1 THEN 580                            7402
560       IF VPEEK(Y)=32 THEN 590                   7B3A
570         FL=1                                    7F0B
580     A$=CHR$(VPEEK(Y))+A$                        8617
590   NEXT Y                                        8839
600   KEY F,A$                                      8B3B
610 NEXT F                                          900A
620 SCREEN 0 : WIDTH 40 : KEY ON                    95BF
630 FOR F=1 TO 4 : KEY(F) OFF : NEXT F              A058
640 PRINT CHR$(27);"x4" : END                       AC51
650 BEEP : BEEP : BEEP                              B228
660 GOSUB 790 : LOCATE,,0                           B976
670 FOR F=0 TO 9                                    BF27
680   FOR G=0 TO 15                                 C738
690     A=PEEK(&HF87F+16*F+G)                       D3F2
700     IF A=0 THEN 720                             DFE4
710       VPOKE40*(F+7)+17+G,A                      EEEA
720   NEXT G                                        F0EE
730 NEXT F                                          F209
740 LOCATE,,1                                       F50E
750 RETURN                                          F6BC
760 BEEP : BEEP : BEEP                              F887
770 DEFUSR=&H3E : S=USR(0)                          145
780 GOSUB 650 : RETURN                              496
790 BEEP : BEEP : BEEP                              98D
800 LOCATE,,0 : PRINT" ";CHR$(8)                    1258
810 FOR F=0 TO 9                                    175F
820   FOR G=0 TO 14                                 1D48
830     VPOKE40*(F+7)+17+G,32                       2858
840   NEXT G                                        2C97
850 NEXT F                                          2F83
860 LOCATE,,1                                       3325
870 RETURN                                          340E

                                 TOTAL   =          340E
```

6.8 - CLS EM MSXDOS

O pequeno programa em BASIC listado a seguir gera um arquivo BATCH que limpa a tela ao ser executado a partir do MSXDOS.

Após digitá-lo, conferí-lo e gravá-lo, rode-o.

Dessa forma, deverá ter sido gerado no disco presente no drive A um arquivo BAT de nome "CLS.BAT". Esse arquivo deve ser usado a partir do MSXDOS, simplesmente digitando CLS e RETURN.

```
10 OPEN "A:CLS.BAT" AS #1
20 FIELD #1, 2 AS A$
30 LSET A$=CHR$(12)+CHR$(26)
40 PUT #1,1
50 END
```

Após rodar o programa, experimente carregar o MSXDOS e comande:

```
A>CLS
```

A tela deverá ser limpa.

6.9 - BEEP EM MSXDOS

O programa em BASIC listado a seguir gera um arquivo BATCH que produz um BEEP quando executado a partir do MSXDOS.

Seu uso é semelhante ao do programa CLS.BAT gerado na dica 6.C e, portanto, ele deve ser executado necessariamente a partir do MSXDOS.

```
10 OPEN "A:BEEP.BAT" AS #1
20 FIELD #1, 2 AS A$
30 LSET A$=CHR$(7)+CHR$(26)
40 PUT #1,1
50 END
```

TOTAL = 1226

Após rodar o programa, carregue o MSXDOS e comande:

```
A>BEEP
```

Isso deverá produzir um beep.

6.A - "SWAP" DE ARQUIVOS EM MSXDOS

Estando em MSXDOS, digite a sequência de comandos listada a seguir para produzir um arquivo BATCH capaz de trocar os nomes de dois arquivos quaisquer.

```
A>COPY CON A:SWAP.BAT
REN %1 BABA.$$$
REN %2 %1
REN BABA.$$$ %2
^Z
```

Com o arquivo SWAP.BAT já gravado no disco, você poderá usá-lo com a seguinte sintaxe:

```
A>SWAP arquivo1 arquivo2
```

Os parâmetros "arquivo1" e "arquivo2" são os nomes dos dois arquivos cujos conteúdos se deseja trocar.

DICAS DE PROCESSAMENTO

As dicas apresentadas neste capítulo são as mais diversas, abordando recursos múltiplos das máquinas MSX. Elas não são específicas para nenhum periférico, sendo mais relacionadas com o processamento de outros programas.

7.1 - Diferenciando o Expert do Hotbit 154
7.2 - Escondendo listagens de programas 155
7.3 - Desativando todos os comandos 157
7.4 - Evitando o "ON STOP GOSUB" 158
7.5 - Obtendo o valor de PI 159
7.6 - Tabelas de caracteres 160
7.7 - Sorteio de palavras da ROM 163
7.8 - Sorteio de palavras em linhas "DATA's" 164
7.9 - Mapeando linhas REM 165
7.A - Mapeando variáveis 166
7.B - Mapeando linhas de um programa 167
7.C - Gerando linhas DATA 168
7.D - Trocando tokens num programa 169
7.E - Aumentando a velocidade de execução ... 171
7.F - "SEARCH", pesquisador de strings 173
7.G - Recuperando programas apagados com NEW 175
7.H - Redefinindo mensagens de erros 176
7.I - Soma sintática 178
7.J - PSEUDO-RAMDISK 181
7.K - Rotacionando caracteres 183
7.L - Rotina de entrada com INKEY$ 184
7.M - Rotina para maiúsculas 186
7.N - Reduzindo a tela a um caractere 187
7.O - Grandes expoentes 188

7.1 - DIFERENCIANDO O EXPERT DO HOTBIT

Existem muitas diferenças facilmente identificáveis entre o HOTBIT e o EXPERT. A mais fácil de ser checada por um programa, quer esteja ele em BASIC ou em Linguagem de Máquina, é a mensagem de identificação do fabricante. No Expert, a partir do endereço 32513 encontramos na ROM a string "G r a d i e n t e". Basta então checar uma letra dessa mensagem que difira do HOTBIT, por exemplo, a letra "G" (código ASCII 71). O programa a seguir faz exatamente isso.

```
100 SCREEN 0 : WIDTH 39 : PRINT                  403
110 PRINT "  Conteúdo da ROM a partir"           E03
120 PRINT "  do endereço 32513: ";               177F
130 FOR F=32513 TO 32531                         1049
140  PRINT CHR$(PEEK(F));                        2500
150 NEXT F                                       2703
160 X$= "é"                                      2030
170 IF PEEK(32513)<>71 THEN X$="não é"           3010
180 PRINT : PRINT : PRINT "  ";                  4554
190 PRINT "Portanto, ";X$;" um EXPERT!"          5B89
200 PRINT : PRINT                                5F05

                           TOTAL   =             5F05
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 82 e 83.

7.2 - ESCONDENDO LISTAGENS DE PROGRAMAS

Um programa em BASIC pode estar presente na memória do micro sem que o comando LIST ou LLIST faça sua listagem aparecer na tela ou na impressora.

A seguir são apresentados dois pequenos programas que exemplificam duas das muitas formas disponíveis para se esconder a listagem.

Em ambos os programas os comandos responsáveis pela ocultação das listagens encontram-se na linha 10.

O primeiro programa deve ser rodado ao menos uma vez antes de ser gravado. Feito isso, o programa deve ser gravado no formato compactado, isto é, não pode ser gravado em ASCII! Para fita cassete deve-se usar o comando CSAVE e para disquete deve-se usar o comando SAVE sem a opção ",A"!

PROGRAMA 1

```
10 POKE&H8003,&HFF:POKE&H8004,&HFF            528
20 FOR F=1 TO 200                             984
30   PRINT "ESTOU AQUI ESCONDIDO !!!"         1384
40 NEXT F                                     1558

                           TOTAL  =           1558
```

O segundo programa pode ser gravado mesmo sem ter sido rodado, pois de qualquer forma, ao ser carregado da fita ou do disquete, sua listagem estará visível. Para que ela se torne inacessível é necessário que ele seja rodado ao menos uma vez após ter sido carregado!

PROGRAMA 2

```
10 POKE&HFF89,&HD1                            346
20 FOR F=1 TO 200                             741
30   PRINT "ESTOU AQUI ESCONDIDO !!!"         1135
40 NEXT F                                     1200

                           TOTAL  =           1200
```

Uma variante mais drástica do programa 2 é obtida alterando-se a linha 10 para:

```
10 POKE &HFF89,&HC3
```

Com isso, toda vez que o comando LIST for

executado, o micro será automaticamente ressetado.
Para desativar essa "trava" basta comandar:

```
POKE &HFF89,HC9
```

Note que apenas o comando LIST foi desabilitado. Os demais comandos do BASCIC continuam funcionando normalmente.

Ao invés de ressetar o micro, podemos simplesmente evitar a listagem. Acrescente ao programa 2 as linhas a seguir:

```
11 POKE &HFF8A,&H72
12 POKE &HFF8B,&H0
```

7.3 - DESATIVANDO TODOS OS COMANDOS

O hook do "Ok" é chamado pelo interpretador sempre que o "Ok" vai ser mostrado na tela. Isso pode servir para evitar que sejam usados quaisquer comandos do BASIC MSX. Para produzir um desvio no hook do "Ok" basta usar os seguintes comandos:

```
POKE &HFF07,&HC3
POKE &HFF08,&H00
POKE &HFF09,&H00
```

Com isso, sempre que o "Ok" for ser impresso na tela, o micro será ressetado. Para voltar a operar normalmente, basta comandar:

```
POKE &HFF07,&HC9
```

Um exemplo prático dessa "trava" pode ser imaginado ao se interromper a execução de um programa em BASIC com CONTROL + STOP. Logo a seguir, o "Ok" surge na tela, provocando um resset automático do micro.

7.4 - EVITANDO O "ON STOP GOSUB"

O MSX permite que se "trave" um programa em BASIC após o início de sua execução com as instruções:

```
STOP ON    e
ON STOP GOSUB xxxxx
```

Usando essas instruções nas primeiras linhas de um programa, após serem executadas, elas farão com que sempre que as teclas CONTROL e STOP forem pressionadas conjuntamente, a execução seja desviada para a linha de número xxxxx.

Isso pode ser muito útil depois que o programa está pronto, mas durante sua elaboração pode trazer transtornos ao programador.

O MSX, porém, possui o veneno e o antídoto. Com apenas um POKE numa variável do sistema é possível acionar o "warm start" do interpretador BASIC pelo teclado. Imagine que vamos executar um programa que usa as instruções STOP ON e ON STOP GOSUB. Se quisermos torná-lo facilmente interrompível, basta comandar:

```
POKE &HFBB0,1
```

Isso fará com que o pressionamento conjunto das teclas CONTROL, SHIFT, LGRA e RGRA no Expert (ou CTRL, SHIFT, GRAPH e CODE no HOTBIT) interrompa a execução do programa e devolva o comando ao usuário.

Para desligar o acionamento do "warm start" do interpretador basta comandar:

```
POKE &HFBB0,0
```

Esse recurso não é exclusivo do BASIC. Mesmo quando um programa em Linguagem de Máquina está sendo executado, se o conteúdo de &HFBB0 for diferente de 0, a execução pode ser interrompida, desde que o programa em L.M. não tenha desabilitado a interrupção!

7.5 - OBTENDO O VALOR DE PI

O MSX não possue uma variável reservada para o armazenamento da constante matemática π.

Entretanto, essa constante pode ser facilmente obtida e de várias formas diferentes.

A forma mais imediata é usar a expressão:

```
4*ATN(1)
```

Experimente comandar:

```
PRINT 4*ATN(1)
```

O valor de π surgirá na tela, pois:

$$\tan(\pi/4) = 1$$

$$\operatorname{Atn}(\tan(\pi/4)) = \operatorname{Atn}(1)$$

$$\pi/4 = \operatorname{Atn}(1)$$

$$\pi = 4*\operatorname{Atn}(1)$$

Outra forma de obter o valor dessa constante é buscá-lo na própria ROM do micro. Os dois programas apresentados a seguir fazem exatamente isso.

```
10 REM PIROM I                              2AD
20 A# = 0                                   4AF
30 FOR F=0 TO 7                             83B
40 POKE VARPTR(A#)+F,PEEK(&H2D43+F)         F6F
50 NEXT F                                   10F4
60 PI = 2*A#                                1488
70 PRINT PI                                 1625

                              TOTAL =       1625
```

```
10 REM PIROM II                             2F2
20 FOR F=0 TO 7                             611
30 A$ = A$ + CHR$(PEEK(&H2D43+F))           EF5
40 NEXT F                                   1077
50 PI = 2*CVD(A$)                           1679
60 PRINT PI                                 1892

                              TOTAL =       1892
```

Após a execução de um desses dois programas, o valor de π ficará armazenado na variável PI.

7.6 - TABELAS DE CARACTERES

O programa apresentado a seguir gera na tela a tabela de caracteres do seu micro. A apresentação é feita em duas partes: a primeira contém os caracteres de 0 a 127 e a segunda os caracteres de 128 a 255.

Além dos próprios caracteres, a tabela contém um quadro em branco logo abaixo de cada um deles. Esses quadros estão divididos em três campos e devem ser preenchidos por você, após a impressão da tabela em papel. Eles servirão para indicar as teclas a serem pressionadas para que o caractere correspondente seja apresentado na tela. Por exemplo, o quadro preenchido abaixo corresponde ao caractere "█".

O campo principal indica a tecla em que o caractere se encontra. O campo superior a direita, quando preenchido, indica que a tecla SHIFT deve ser pressionada. O campo inferior direito, quando preenchido, indica que a tecla LGRA (ou GRAPH) deve ser pressionada. Portanto, para produzir o caractere "█" deve-se pressionar as teclas:

SHIFT + LGRA + 9 (**Expert**)
ou
SHIFT + GRAPH + 9 (**Hotbit**)

O campo inferior direito serve para indicar se a tecla RGRA (ou CODE) deve ser usada. Nesse caso, ele deve ser preenchido.

Caso o caractere não possa ser produzido através do teclado, sugerimos que não se preencha nada no campo reservado para a tecla. Cabe aqui uma crítica aos fabricantes: alguns caracteres não acessáveis pelo

teclado são símbolos reservados de algumas linguagens de programação. O caractere "til", por exemplo, é usado como operador lógico de negação para bits na linguagem C. Isso faz parte do padrão da linguagem e o próprio compilador MSX-C, da ASC Corporation, o adota. O usuário de um EXPERT ou de um HOTBIT, entretanto, terá que fazer algumas "mágicas" antes de conseguir usar esse operador lógico num programa em C. Outros caracteres, como o "ô", por exemplo, são repetidos desnecessariamente, uma vez que o caractere maiúsculo e o minúsculo têm o mesmo desenho.

```
100 '
110 ' TABLE DE CARACTERES
120 '
130 SCREEN 2,,,,1
140 OPEN "GRP:" AS #1
150  GOSUB 370      :' horizontais
160  GOSUB 460      :' verticais
170  GOSUB 560      :' tracinhos
180  H = 0
190  GOSUB 250      :' caracteres
200  A$ = INPUT$(1)
210  H = 128
220  GOSUB 250      :' caracteres
230  A$ = INPUT$(1)
240  GOTO 180
250 '
260 '   desenha caracteres
270 '
280 FOR F=H TO H+127
290  L = (F-H)\16
300  C = 4+F-H-16*L
310  LINE(12*C,13+22*L)-
       (12*C+7,20+22*L),1,BF
320  PSET (12*C,13+22*L),1
330  IF F<32 THEN PRINT #1,CHR$(1)+
                          CHR$(64+F)
340  IF F>31 THEN PRINT #1,CHR$(F)
350 NEXTF
360 RETURN
370 '
380 '   linhas horizontais
390 '
400 FOR F=0 TO 187 STEP 11
410   IF F/2<>F\2 THEN LINE(33,F)
                          -(45,F)
420   LINE(45,F)-(238,F)
```

```
430    IF F/22=F\22 THEN LINE(45,F+10)
                         -(238,F+10)
440 NEXT F
450 RETURN
460 '
470 '    linhas verticais
480 '
490 FOR F=33 TO 238 STEP 12
500  IF F>33 THEN LINE (F,0)-(F,12)
510  LINE(F,12)-(F,187)
520  IF F>33 THEN LINE(F+1,12)-
                    (F+1,187)
530 NEXT F
540 LINE (F-11,0)-(F-11,12)
550 RETURN
560 '
570 '    desenha  T
580 '
590 FOR F=45 TO 226 STEP 12
600  FOR G=22 TO 176 STEP 22
610   LINE(F+6,G)-(F+6,G+10)
620   LINE(F+6,G+4)-(F+12,G+4)
630   LINE(F+9,G+4)-(F+9,G+10)
640  NEXT G
650 NEXT F
660 RETURN
```

7.7 - SORTEIO DE PALAVRAS DA ROM

O programa apresentado a seguir sorteia uma palavra reservada do BASIC MSX a partir da digitação de alguma tecla pelo usuário. Você pode usá-la em outros programas para obter palavras ao acaso.

```
100 '  rotina de sorteio                          934
110 A$=INPUT$(1) : BEEP                           1071
120 A=65536!*RND(-TIME)                           1521
130 A%=163*RND(A)+1                               1081
140 EN=14962 : A$="" : I=65 : C=0                 27EE
150 A$=A$+CHR$(I)                                 2086
160 P=PEEK(EN) : Q=PEEK(EN+1)                     384B
170 P$=CHR$(P)                                    3DB7
180 IF P<128 THEN A$=A$+P$ : GOTO 240             4C06
190 A$=A$+CHR$(P-128)                             5425
200 EN=EN+1 : C=C+1                               5EA3
210 IF C=A% THEN PRINT A$ : RUN                   6B20
220 IF PEEK(EN+1)=0 THEN 240                      76A4
230   A$="" : A$=A$+CHR$(I)                       83F0
240 IF PEEK(EN)<>0 THEN 280                       8B99
250 A$="" : I=I+1 : Q$=CHR$(I)                    944B
260 IF Q$="J" OR Q$="Q" THEN 280                  9D44
270 A$=A$+Q$                                      A0EB
280 EN=EN+1                                       A3BB
290 IF EN<=15649 THEN 160                         AB27

                              TOTAL   =           AB27
```

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.2 - páginas 50 a 55.

7.8 - SORTEIO DE PALAVRAS EM LINHAS "DATA'S"

O programa apresentado abaixo sorteia uma palavra entre uma coleção delas inseridas em linhas datas. Experimente acrescentar ao programa outras linhas DATA's com mais palavras. Note que é essencial que a última palavra seja "FIM" para que o programa pare de tentar ler mais palavras.

```
100 '  rotina de sorteio                         934
110 A$=INPUT$(1) : BEEP :RESTORE                 E2F
120 A=65536!*RND(-TIME) : F%=0                   1790
130 READ A$ : IF A$="FIM" THEN 150               2109
140            F%=F%+1 : GOTO  130               2006
150 A%=F%*RND(A)+1 : RESTORE                     3733
160 FOR F%=1 TO A%                               3E4E
170   READ A$                                    4304
180 NEXT F%                                      4509
190 PRINT A$                                     48CE
200 GOTO 110                                     4C8F
1000 REM                                         4E04
1010 DATA BABA,CACA,DADA,FAFA,GAGA,HAHA          63CA
1011 DATA JAJA,KAKA,LALA,MAMA,NANA,PAPA          79C8
1012 DATA PAPA,RARA,SASA,TATA,UAUA,VAVA          888A
1013 DATA XAXA,ZAZA,NICK,GUTT,MINHOLETA          9332
1020 DATA FIM                                    966F

                           TOTAL   =             966F
```

7.9 - MAPEANDO LINHAS REM

O programa apresentado a seguir vasculha a memória RAM do micro a partir de &H8000 a procura de pseudo-instruções REM. Ao encontrar no programa um REM (ou '), a linha em que ele se encontra terá seu número apresentado na tela.

Após digitá-lo, grave-o no formato ASCII e com a numeração bem alta (como usamos a seguir). Para usá-lo, faça um MERGE com o programa que estiver na memória e comande:

```
RUN 65100
```

```
65100 REM                                          105
65110 REM PROCURA LINHAS REM                       97A
65120 REM                                          B6C
65130 PRINT CHR$(12)                               F51
65140 EI = 32769!                                  13AC
65150 B1 = PEEK (EI)                               1944
65160 B2 = PEEK (EI + 1)                           217E
65170 B3 = PEEK (EI + 2)                           296E
65180 B4 = PEEK (EI + 3)                           3110
65190 PL = B1 + 256*B2                             39E8
65200 NL = B3 + 256*B4                             44BC
65210 IF B1=0 AND B2=0 THEN 350                    501C
65220  PRINT"================================      684C
"
65230  PRINT"  NUL       :"; NL                    77E8
65240  PRINT"  CPL       :"; PL - EI               89E4
65250  PRINT"  EPL       :";"&H";HEX$(PL)          92C0
65260  FOR F = (EI + 4) TO (PL - 2)                9BE8
65270   IF PEEK(F)<>&H8F THEN 310                  A610
65280    PRINT,"&H";HEX$(F);" => REM";             B0E2
65290    PRINT CHR$(7)                             B6BC
65300    F = PL - 2                                BDF5
65310  NEXT F                                      C12E
65320  PRINT"================================      D54D
"
65330  EI=PL                                       DAF1
65340  GOTO 150                                    DEC1
65350 END                                          E03B

                          TOTAL  =                 E03B
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.A - MAPEANDO VARIÁVEIS

O programa apresentado a seguir vasculha a memória RAM do micro a partir de &H8000 a procura de uma dada variável. Se ela existir no programa, a linha em que ela se encontra terá seu número apresentado na tela.

Após digitá-lo, grave-o no formato ASCII e com a numeração bem alta (como usamos a seguir). Para usá-lo, faça um MERGE com o programa que estiver na memória e comande:

```
RUN 65000
```

```
65000 REM                                          274
65001 REM MAPA DE VARIAVEIS                        9ED
65002 REM                                          C5B
65003 PRINT CHR$(12)                               1100
65004 POKE &HFCAB,1                                1510
65005 INPUT"VARIAVEL";V$                           199E
65006 EI = 32769!                                  209E
65007 B1 = PEEK(EI)   : B2 = PEEK(EI + 1           2EEB
)
65008 B3 = PEEK(EI+2) : B4 = PEEK(EI + 3           3E72
)
65009 PL = B1 + 256*B2 : NL = B3 + 256*B           5505
4
65010 IF B1=0 AND B2=0 THEN 65024                  6182
65011  IF PEEK(EI+4)=&H8F THEN 65022               6D07
65012  IF PEEK(EI+5)=&H8F THEN 65022               7997
65013   L$="" : IP=0                               8218
65014   FOR F = (EI+4) TO (PL-2)                   8DBE
65015    IF PEEK(F)=34 THEN IP=(IP+1)              95F8
65016    IF IP/2<>IP\2 THEN 65018                  9F6B
65017     L$ = L$ + CHR$(PEEK(F))                  A868
65018   NEXT F                                     AE8E
65019   L=INSTR(L$,V$)                             B3CE
65020   IF L=0 THEN 65022                          B894
65021   PRINT "))";STR$(NL)+" ((",                 C95A
65022  EI=PL                                       CE84
65023  GOTO 65007                                  D1E3
65024 POKE &HFCAB,0                                D5F0
65025 END                                          D781

                              TOTAL   =            D781
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.B - MAPEANDO LINHAS DE UM PROGRAMA

O pequeno programa listado a seguir mapeia as linhas de um programa em BASIC presente na memória do micro. Ele deve ser gravado em ASCII para poder ser emendado (com o MERGE) ao final dos programas que se deseja mapear.

Para usá-lo, após ter feito o MERGE, basta comandar:

```
RUN 60000
```

Os dados serão apresentados em 4 colunas. A 1ª conterá o nº da linha; a 2ª conterá seu endereço inicial; a 3ª seu endereço final e a 4ª o comprimento da linha em bytes.

```
60000 REM MAPEADOR DE LINHAS                        75F
60010 EPL=32769!                                    C40
60020 PRINT"LINHA  E.I.    E.F       Comp.          151A
"
60030 PRINT"-------------------------------         2364
"
60040 NL=PEEK(EPL+2)+256*PEEK(EPL+3)                2F7D
60050 PRINT NL;"=> ";HEX$(EPL);" a ";               4091
60060 C=EPL+1:EPL=PEEK(EPL)+256*PEEK(C)             535C
60070 PRINT HEX$(EPL-1);" => ";EPL-C+1              644F
60080 IF PEEK(EPL)=0 THEN END                       6053
60090 GOTO 60040                                    7020

                              TOTAL   =             7020
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.C - GERANDO LINHAS DATA

Muitas vezes nos deparamos com situações em que necessitamos armazenar os bytes de uma região da memória em linhas DATA. O programa apresentado a seguir gera na tela, na impressora, em disquete ou em cassete, as linhas DATA correspondentes a uma certa região da memória.

Ao ser executado o programa pede a introdução do número da primeira linha DATA a ser gerada e os endereços inicial e final da área da memória a ser lida.

Na forma como está o programa a seguir, a listagem das linhas DATAS é apresentada na tela. Para fazer com que ela seja enviada à impressora, ao drive ou ao cassete, deve-se alterar a linha 130, substituindo o nome "CRT:" pelo do dispositivo desejado.

```
100 REM GERADATA2
110 REM
120 CLEAR 1000
130 OPEN"CRT:DATAS" FOR OUTPUT AS #1
140 DEFSNG A-Z
150 L$ = ""
160 INPUT"Entre a linha inicial";L
170 IF L<0 OR L>65000! THEN 160
180 INPUT"Entre o endereço inicial";I$
190 I = VAL("&H"+I$)
200 IF I<&H8000 OR I>&HFFFF THEN 190
210 INPUT"Entre o endereço final";F$
220 F = VAL("&H"+F$)
230 IF F<I OR F>&HFFFF THEN 220
240 L$ = L$ + STR$(L)+" DATA "
250 FOR N=I TO F
260   X$ = "00" + HEX$(PEEK(N))
270   X$ = RIGHT$(X$,2)
280   X = X + 1
290   L$ = L$ + X$
300   IF X/8<>X\8 THEN 350
310     PRINT #1,L$
320     L = L + 10
330     L$ = STR$(L)+" DATA "
340     GOTO 360
350   L$ = L$ + ","
360 NEXT N
370 IF RIGHT$(L$,1)<>"," THEN END
380 L$ = LEFT$(L$,LEN(L$)-1)
390 PRINT #1,L$
```

7.D - TROCANDO TOKENS NUM PROGRAMA

Enfrentar o problema de transformar todos os comandos de um programa em outro comando não é uma tarefa das mais agradáveis, principalmente quando o programa é grande.

A rotina a seguir coloca um programa em Linguagem de Máquina a partir do endereço &HC000. Quando executada ela varre a área do programa em BASIC trocando os códigos iguais ao contido no endereço &HC09C pelo código contido em &HC09B.

Você pode usar o programa abaixo como uma rotina em BASIC, usando-o com um "MERGE", ou gravá-lo em binário e "pokear" os códigos da tokens manualmente.

Os códigos de todas as tokens pode ser encontrados no livro "APROFUNDANDO-SE NO MSX", na páginas 67 a 71 ou usando o programa da figura 1.4 do livro "PROGRAMAÇÃO AVANÇADA EM MSX", nas páginas 11 e 12.

Lembre-se que as tokens de funções tem sempre o bit 7 setado antes de serem inseridas num programa, portanto, se você desejar substituir tokens de funções deverá antes acrescentar &H80 ao seu código. Por exemplo, para substituir a token da função seno (&H09) pela da função cosseno (&H0C) você deverá "pokear" os valores &H89 e &H8C respectivamente nos endereços &HC09C e &HC09B.

```
50000 '*********************
50010 '* BY THE DOCTOR LUZ *
50020 '*********************
50030 FOR L=&HC000 TO &HC09F
50040 READ A$:POKE L,VAL("&H"+A$)
50050 NEXT
50060 DEF USR=&HC000
50070 PRINT"RODAR PROGRAMA ?"
50080 A$=INPUT$(1)
50090 IF A$="S" THEN GOTO 50100 ELSE END
50100 INPUT"CODIGO DA TOKEN";A
50110 INPUT"NOVO CODIGO";B
50120 POKE &HC09C,A
50130 POKE &HC09B,B
50140 A=USR(0)
50150 LIST
50160 DATA DD,21,FF,7F,DD,23,21,73
50170 DATA C0,DD,7E,00,06,14,BE,28
50180 DATA 21,23,23,10,F9,DD,7E,00
50190 DATA FE,22,28,39,5F,3A,9C,C0
50200 DATA DD,BE,00,20,DF,3A,9B,C0
```

```
50210 DATA DD,77,00,7B,FE,84,28,36      AFA9
50220 DATA 18,D2,DD,7E,00,FE,00,20      C059
50230 DATA 0E,DD,7E,01,FE,00,20,07      1C8E
50240 DATA DD,7E,02,FE,00,28,09,23      EA11
50250 DATA 7E,47,DD,23,10,FC,18,B4      F928
50260 DATA DD,22,9D,C0,C9,DD,23,DD      5E
50270 DATA 7E,00,FE,16,28,A6,FE,00      627
50280 DATA 20,F3,DD,2B,18,9E,DD,23      F8E
50290 DATA DD,7E,00,FE,00,20,F7,DD      1783
50300 DATA 2B,18,91,00,04,0B,03,0C      2708
50310 DATA 03,0D,03,0E,03,0F,02,11      3745
50320 DATA 01,12,01,13,01,14,01,15      47AF
50330 DATA 01,16,01,17,01,18,01,19      5E7C
50340 DATA 01,1A,01,1C,03,1D,04,1F      7251
50350 DATA 08,FF,02,00,00,00,00,20      78A6

                              TOTAL  =  78A6
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 2.
Programação Avançada em MSX - capítulo 1.

7.E - AUMENTANDO A VELOCIDADE DE EXECUÇÃO EM BASIC

Para ilustrar alguns métodos de se aumentar a velocidade de execução de programas em BASIC vamos usar uma pequena rotina que calcula os números primos compreendidos entre 2 e 1000. Rode o programa a seguir e anote o tempo que ele levou para rodar.

```
100 TIME=0                                      22E
120 DIM A(2000)                                 4F4
130 FOR I=2 TO 1000                             9FF
140  IF A(I)<>0 THEN GOTO 190                   10EF
150   PRINT I : P=P+1                           1608
160   FOR N=2*I TO 2000 STEP I                  1F20
170     A(N)=1                                  2491
180   NEXT N                                    27A4
190 NEXT I                                      2A3C
200 PRINT : PRINT "TEMPO:";TIME/60              331F
210 END                                         33F7

                              TOTAL =           33F7
```

O MSX trabalha normalmente com números em dupla precisão (14 casas) e faz os cálculos em BCD para reduzir ao máximo os erros de arredondamento. Com isso, ele perde mais tempo que outros micros menos inteligentes, como os APPLE's, os TRS-80, etc. Mas podemos fazer com que ele use precisão simples (6 casas), uma vez que as operações envolvidas são apenas a adição e a multiplicação de números inteiros. Isso faz o programa ficar ligeiramente mais rápido. Experimente acrescentar ao programa anterior a linha a seguir, rode-o novamente e anote o tempo de execução.

```
110 DEFSNG A-Z
```

Em nosso caso específico, sabemos que os dados manipulados pelo programa são todos inteiros. Todos os números primos são inteiros. Sabendo disso, podemos fazer com que o MSX diminua a precisão de seu cálculos um pouco mais. Altere a linha 110 como mostrada a seguir e rode mais uma vez o programa. Você deverá obter um tempo de execução bem menor.

```
110 DEFINT A-Z
```

Isso ainda não é tudo. Existem alguns procedimentos que o micro faz ao executar pela primeira vez uma linha do programa que não são refeitos se ela for executada outras vezes. Altere o programa, deixando-o como mostrado a seguir e rode-o mais uma vez.

```
0  TIME=0:KEYOFF:LOCATE0,0,0
1 DEFINTA-Z:POKE&HF3B1,3:DIMA(2000)
2 FORI=2TO100
3 IFA(I)<>0THEN7
4 PRINTI:P=P+1
5 FORN=2*ITO2000STEPI
6 A(N)=1:NEXT
7 NEXT
8 PRINT:PRINT"TEMPO:";TIME/60
9 END
```

Além da renumeração, da supressão dos espaços em branco e da compactação de várias instruções por linha, o programa foi otimizado com mais algumas instruções.

A linha 0, além de apagar as teclas de funções, apaga definitivamente o cursor da tela.

Na linha 1, a instrução POKE na posição &HF3B1 faz com que a tela tenha apenas 3 linhas! Isso reduz bastante o tempo de 'SCROLL' da tela quando ela já está cheia.

Por fim, as instruções NEXT sem especificação do parâmetro nas linhas 6 e 7 também reduzem um pouco o tempo de execução.

Se você tiver um APPLE, um TRS-80 ou mesmo um PC, experimente rodar neles um programa equivalente e compare o tempo gasto com o do MSX. A não ser no caso de um PC com clock de 8 Mhz, o MSX será o mais rápido!

BIBLIOGRAFIA RECOMENDADA:

Curso de Basic MSX v.1 - páginas 37 a 39.
Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.F - "SEARCH", PESQUISADOR DE STRINGS

O programa apresentado a seguir gera, a partir do endereço &HE000, uma rotina em Linguagem de Máquina que pode ser chamada pelo BASIC para vasculhar a memória a procura de uma sequência de caracteres (string). Após gerar a rotina em L.M., o programa a grava em fita ou em disco com o nome SEARCH ou "SEARCH.BIN", respectivamente.

Para testar o programa, após ele ter sido executado, comande:

```
PRINT,HEX$(USR0("A L E P H"))
```

Ao fazer isso, a string "A L E P H" será procurada na memória e o endereço em que ela for encontrada será mostrado em hexadecimal na tela.

Após ter salvo o programa em BASIC, você pode apagá-lo da memória que mesmo assim a rotina em L.M. permanecerá funcionando. Para usá-la a sintaxe é sempre a mesma: basta chamar a função USR0 passando como parâmetro a string a ser procurada.

```
10 FOR F%=&HE000 TO &HE14B
20  READ A$ : POKE F%,VAL("&H"+A$)
30 NEXT F% : DEFUSR0=&HE000
40 BSAVE"SEARCH.BIN",&HE000,&HE14B
50 END
100 DATA FE,03,C0,D5,11,4D,E0,21
110 DATA 4C,E0,36,00,01,FF,00,ED
120 DATA B0,D1,06,00,1A,4F,C5,13
130 DATA 1A,6F,13,1A,67,11,4C,E0
140 DATA ED,B0,21,00,00,01,FF,FF
150 DATA 11,4C,E0,1A,ED,B1,28,01
160 DATA C9,13,1A,B7,28,09,ED,A1
170 DATA E0,28,F6,1B,03,18,E9,AF
180 DATA C1,ED,42,22,F8,F7,3E,02
190 DATA 32,63,F6,C9,41,42,56,41
200 DATA 42,56,53,00,00,00,00,00
210 DATA 00,00,00,00,00,00,00,00
220 DATA 00,00,00,00,00,00,00,00
230 DATA 00,00,00,00,00,00,00,00
240 DATA 00,00,00,00,00,00,00,00
250 DATA 00,00,00,00,00,00,00,00
260 DATA 00,00,00,00,00,00,00,00
270 DATA 00,00,00,00,00,00,00,00
280 DATA 00,00,00,00,00,00,00,00
290 DATA 00,00,00,00,00,00,00,00
300 DATA 00,00,00,00,00,00,00,00
```

```
310 DATA 00,00,00,00,00,00,00,00   3EC8
320 DATA 00,00,00,00,00,00,00,00   4EBD
330 DATA 00,00,00,00,00,00,00,00   66AF
340 DATA 00,00,00,00,00,00,00,00   7BCA
350 DATA 00,00,00,00,00,00,00,00   85A6
360 DATA 00,00,00,00,00,00,00,00   8997
370 DATA 00,00,00,00,00,00,00,00   8F95
380 DATA 00,00,00,00,00,00,00,00   9AB4
390 DATA 00,00,00,00,00,00,00,00   A454
400 DATA 00,00,00,00,00,00,00,00   B631
410 DATA 00,00,00,00,00,00,00,00   C40B
420 DATA 00,00,00,00,00,00,00,00   D95E
430 DATA 00,00,00,00,00,00,00,00   F012
440 DATA 00,00,00,00,00,00,00,00   153
450 DATA 00,00,00,00,00,00,00,00   6E1
460 DATA 00,00,00,00,00,00,00,00   A18
470 DATA 00,00,00,00,00,00,00,00   1620
480 DATA 00,00,00,00,00,00,00,00   2045
490 DATA 00,00,00,00,00,00,00,00   2EA7
500 DATA 00,00,00,00,00,00,00,00   3F12
510 DATA 00,00,00,00,00,00,00,00   4EBE
```

Experimente, após ter rodado o programa ao menos uma vez, desligar o micro, ligá-lo novamente, carregar o programa SEARCH gravado em binário para a memória do micro e comandar:

```
DEFUSR0=&HE000 : ? HEX$(USR0("color"))
```

Você deverá obter como resultado o endereço da ROM onde existe essa palavra.

7.G - RECUPERANDO PROGRAMAS APAGADOS COM NEW

O programa apresentado a seguir deve ser digitado e gravado. Ao ser executado ele gera e grava um programa em Linguagem de Máquina capaz de recuperar programas em BASIC apagados da memória do micro com o comando NEW.

Existem alguns raros casos em que a rotina não funcionará, porém certamente esse casos constituem menos de 1% das situações reais.

Para testar o programa, após tê-lo digitado, gravado e executado, comande NEW.

O programa foi "apagado" da memória. NÃO FAÇA NADA AINDA para evitar perder dados que ainda estão na memória do micro (apesar de ele não saber disso)! A primeira coisa a ser feita nessa situação é carregar o programa em Linguagem de Máquina gravado pelo programa em BASIC que você digitou. Comande:

```
BLOAD"WENNEW.BIN",R
```

Isso será o suficiente para recuperar seu programa em BASIC.

```
100 FOR F=&HE000 TO &HE032                    588
110   READ A$:POKE F,VAL("&H"+A$)             C98
120 NEXT F                                    E2F
130 BSAVE"WENNEW.BIN",&HE000,&HE032           177C
140 END                                       1950
150 DATA F3,21,04,80,23,7E,FE,00              241E
160 DATA 20,FA,23,23,7E,FE,80,2B              2CE0
170 DATA 20,F2,22,01,80,EB,1A,6F              4059
180 DATA 13,1A,67,7E,FE,00,20,F5              5130
190 DATA 23,22,C2,F6,22,C4,F6,22              642E
200 DATA C6,F6,21,01,80,22,76,F6              7987
210 DATA FB,C9,FF,00,00,00,00,00              8891

                                 TOTAL =      8891
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.H - REDEFININDO MENSAGENS DE ERROS

O programa apresentado a seguir permite a redefinição das mensagens de erro do BASIC. Na verdade, ele acrescenta novas mensagens às já existentes. Apenas a título de exemplo, usamos mensagens do tipo "MENSAGEM 1", "MENSAGEM 2", etc. Entretanto você pode criar suas próprias mensagens desde que sempre as termine pelo caractere "#".

```
100 REM                                      F3
110 REM ERROS ALTERNATIVOS                   780
120 REM                                      890
130 CLEAR 200,&HD000                         C44
140 FOR F=0 TO 61                            1099
150 READ A$ : A=VAL("&H"+A$)                 180A
160 POKE &HD000+F,A                          1002
170 NEXT F                                   207F
180 READ A$                                  2384
190 IF A$="FIM" THEN 250                     2AB2
200 FOR G=1 TO LEN(A$)                       32AE
210 POKE &HD000+F,ASC(MID$(A$,G,1))          3FC1
220 F=F+1                                    43C7
230 NEXT G                                   4695
240 GOTO 180                                 4956
250 POKE &HD000+F,255                        4F88
260 DEFUSR=&HD000                            53F3
270 S=USR(0)                                 5909
280 BSAVE"ERROS",&HD000,&HD000+F             6592
290 END                                      67C8
300 REM                                      6A0D
310 REM DADOS                                7946
320 REM                                      7240
330 DATA 11,0D,D0,21,FD,FE,36,C3             81F7
340 DATA 23,73,23,72,C9,E5,F5,C5             8794
350 DATA 21,3D,D0,7E,23,FE,FF,28             801C
360 DATA 16,FE,23,20,F6,0D,20,F3             96F3
370 DATA 7E,23,FE,FF,28,09,FE,23             9E0E
380 DATA 28,05,CD,A2,00,18,F1,3E             AE14
390 DATA 0D,CD,A2,00,3E,0A,CD,A2             C02B
400 DATA 00,C1,F1,E1,C9,23                   CCE7
410 DATA "MENSAGEM 1              #"         E1C4
420 DATA "MENSAGEM 2              #"         F617
430 DATA "MENSAGEM 3              #"         50
440 DATA "MENSAGEM 4              #"         563
450 DATA "MENSAGEM 5              #"         E58
460 DATA "MENSAGEM 6              #"         1773
470 DATA "MENSAGEM 7              #"         1FEC
480 DATA "MENSAGEM 8              #"         333F
```

```
490 DATA "MENSAGEM 9                #"    449C
500 DATA "MENSAGEM A                #"    5AE4
510 DATA "MENSAGEM B                #"    6E44
520 DATA "MENSAGEM C                #"    807D
530 DATA "MENSAGEM D                #"    870C
540 DATA "MENSAGEM E                #"    8C8D
550 DATA "MENSAGEM F                #"    941C
560 DATA "MENSAGEM G                #"    9B4D
570 DATA "MENSAGEM H                #"    A764
580 DATA "MENSAGEM I                #"    B9CD
590 DATA "MENSAGEM J                #"    C02C
600 DATA "MENSAGEM K                #"    E105
610 DATA "MENSAGEM L                #"    F604
620 DATA "MENSAGEM M                #"    25
630 DATA "MENSAGEM N                #"    544
640 DATA "MENSAGEM O                #"    EB5
650 DATA "MENSAGEM P                #"    175C
660 DATA "FIM"                            1AE3

                          TOTAL   =       1AE3
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 71 a 76 e 82.

7.1 - SOMA SINTÁTICA

Os erros mais frequentes cometidos durante a transcrição de programas listados para a memória do computador são devidos a digitação incorreta por parte do leitor. Raras as vezes em que o erro é devido a falhas de impressão e mais raras ainda as ocasiões em que o programa está realmente com algum erro lógico.

Mesmo considerando o fato de que todos os programas apresentados neste livro estão em BASIC são, em sua maioria, curtos e de fácil digitação, a probabilidade de falhas durante a digitação é algo considerável. Uma simples vírgula substituida involuntariamente por um ponto pode por todo um programa a perder.

Pensando em como diminuir a ocorrência de erros de digitação desenvolvemos uma pequena rotina em Linguagem de Máquina capaz de checar o programa na memória. Antes de continuarmos, digite o programa a seguir exatamente como o listamos abaixo, sem tirar nem por absolutamente nada e tomando o máximo cuidado para não cometer erro algum. Depois, salve o programa em disco ou em fita cassete.

```
100 REM-----------------------------
110 REM SOMA SINTÁTICA
120 REM                   Rubens Jr.
130 REM-----------------------------
140 FOR F=&HE000 TO &HE0B5
150   READ A$ : POKE F,VAL("&H"+A$)
160 NEXT F : DEFUSR0=&HE000
170 BSAVE"SOMSIN.BIN",&HE000,&HE0B5
180 SS = USR0(0) : END
190 REM-----------------------------
200 DATA 3E,00,32,B4,E0,21,00,00
210 DATA 22,B0,E0,2A,76,F6,CD,F9
220 DATA 10,CD,63,E0,7A,B3,CA,4B
230 DATA E0,ED,53,B2,E0,CD,63,E0
240 DATA CD,68,E0,2B,2B,CD,7E,E0
250 DATA 23,E5,ED,5B,B2,E0,1B,B7
260 DATA ED,52,E1,20,F0,2A,B0,E0
270 DATA CD,94,E0,3E,0D,CD,A2,00
280 DATA 3E,0A,CD,A2,00,2A,B2,E0
290 DATA C3,0E,E0,21,A3,E0,CD,78
300 DATA 66,2A,B0,E0,CD,94,E0,3E
310 DATA 02,32,63,F6,2A,B0,E0,22
320 DATA F8,F7,C9,5E,23,56,23,C9
330 DATA E5,EB,CD,12,34,3E,09,CD
340 DATA A2,00,3E,3D,CD,A2,00,3E
```

```
350 DATA 09,CD,A2,00,E1,C9,5E,16           3AA5
360 DATA 00,3A,B4,E0,3C,32,B4,E0           48FC
370 DATA AB,5F,E5,2A,B0,E0,19,22           6189
380 DATA B0,E0,E1,C9,22,F8,F7,3E           7661
390 DATA 02,32,63,F6,CD,22,37,CD           8568
400 DATA 78,66,C9,0D,0A,0D,0A,54           887E
410 DATA 4F,54,41,4C,09,3D,09,00           8FA5
420 DATA 00,00,00,00,00,42,41,4F           9896

                            TOTAL  =       9896
```

Com o programa já gravado, comande RUN. Você deverá obter na tela uma listagem terminada com a mensagem:

TOTAL = 9896

Se isso não ocorreu, carregue o programa SOMA SINTÁTICA (previamente gravado) para a memória do micro e confira-o novamente, pois há alguma coisa errada nele.

Após obter o valor correto para a soma TOTAL do programa, grave-o definitivamente.

Um programa em BASIC é armazenado na memória do micro como uma sequência de bytes. A rotina que gera a SOMA SINTÁTICA vai lendo a memória e para cada linha do programa em BASIC calcula uma soma "ponderada" dos bytes que a constituem. O valor com que cada byte participa na soma de cada linha depende de seu próprio valor e da posição que ele ocupa na linha. Para verificar isso, comande NEW e introduza a linha a seguir:

```
10 PRINT "AMOR"
```

A seguir, comande:

```
SS = USR0(0)
```

Você deverá obter a soma `TOTAL  =  245`

Agora, altere a linha 10 para:

```
10 PRINT "ROMA"
```

Comande mais uma vez:

```
SS = USR0(0)
```

A soma TOTAL se alterou para 238 e isso aconteceu apesar de as duas linhas terem exatamente os mesmos caracteres.

Quando o programa tem várias linhas, o valor total de cada uma é adicionado na soma da linha subsequente. Desse modo, a soma total de um programa é sempre igual a soma indicada em sua última linha e quando um erro é cometido numa determinada linha, as somas de todas as linhas subsequentes são alteradas.

Note, porém, que a rotina não é infalível. Calcule, como contra-exemplo, as somas TOTAIS das duas linhas abaixo:

```
10 PRINT"ACD"
```

e

```
10 PRINT"ABC"
```

Apesar de as linhas serem diferentes, suas SOMAS são iguais!

Quase todos os demais programas deste livro estão acompanhados de sua soma TOTAL, obtida através do programa SOMA SINTÁTICA rodando num MSX EXPERT versão 1.1. Se o seu MSX for de outro tipo, a soma TOTAL dos programas poderá resultar diferente da que apresentamos, pois alguns caracteres (como o Ç, por exemplo) podem ter códigos diferentes. Cuidado, portanto, se esse for o seu caso!

Outra situação peculiar ocorre com programas que possuem GOTO ou GOSUB. Se calcularmos a soma total desses programas antes de executá-los e depois de executá-los obteremos valores diferentes, pois o interpretador BASIC da ROM do MSX usa uma técnica de otimização que altera as linhas com GOTO ou GOSUB após a primeira passada por elas (veja as páginas 16 e 17 do livro PROGRAMAÇÃO AVANÇADA EM MSX para maiores detalhes sobre essa técnica).

Neste livro, as somas TOTAIS apresentadas foram sempre calculadas ANTES de se executar o programa sequer uma única vez!

7.J - PSEUDO-RAMDISK

O programa apresentado a seguir permite o uso dos 32 Kbytes de RAM não disponíveis para o BASIC (entre os endereços 0 e &H8000). Após digitá-lo e gravá-lo, certificando-se de que ele esteja correto, rode-o. Com isso uma rotina em Linguagem de Máquina estará pronta para ser usada com outros programas, desde que ele não se sobreponham à área de memória entre &HD000 e &HD076.

A rotina permite que os programas presentes na RAM disponível do micro sejam passados para a RAM oculta e posteriormente recuperados de volta para a RAM disponível.

Para passar da RAM ativa para a RAM oculta deve-se comandar:

```
POKE 0,USR0(0%)
```

Fazendo isso você pode carregar outro programa na memória e usá-lo normalmente, pois o programa anterior estará "salvo" na RAM oculta.

Para recuperar o programa da RAM oculta para a RAM ativa deve-se comandar:

```
POKE 0,USR0(1%)
```

Com isso o programa "salvo" estará novamente presente na RAM ativa do micro.

Além desses recursos podemos também "trocar" o conteúdo da RAM ativa e da RAM oculta. Se temos um programa PROGR1 "salvo" na RAM oculta e um outro programa PROGR2 presente da RAM ativa, para trocá-los de posições devemos comandar:

```
POKE 0,USR0(2%)
```

Faça alguns testes com programas curtos para se habituar aos comandos.

```
100 SCREEN 0:WIDTH 39:CLEAR 200,&HD000      595
110 FOR F=&HD000 TO &HD076                  BF8
120   READ A$:POKE F,VAL("&H"+A$)           128E
130 NEXT F                                  1408
140 DEFUSR0=&HD000                          1980
150 DATA F3,F5,C5,D5,E5,ED,73,FE            2291
160 DATA F3,FE,02,20,5E,23,23,7E            2858
170 DATA FE,00,20,0B,21,00,80,11            3054
180 DATA 00,00,CD,53,D0,18,43,FE            4009
```

```
190 DATA 01,20,0B,21,00,00,11,00     6207
200 DATA 80,CD,53,D0,18,34,FE,02     [illegible]
210 DATA 20,39,CD,53,D0,01,FF,5F     8B3D
220 DATA 11,00,00,21,00,80,1A,32     3E00
230 DATA 75,D0,7E,12,3A,75,D0,77     9287
240 DATA 23,13,0B,78,B1,20,EF,DB     9C52
250 DATA A8,18,14,DB,A8,47,CB,3F     A511
260 DATA CB,3F,CB,3F,CB,3F,80,D3     B350
270 DATA A8,C9,01,FF,5F,ED,B0,E6     C5FE
280 DATA F0,D3,A8,ED,7B,FE,F3,E1     1944
290 DATA D1,C1,F1,FB,C9,00,00,00     EF43
```

Você pode também programar as teclas de funções com os comandos de SALVAR, RECUPERAR e TROCAR programas na RAM oculta para facilitar a operação do programa em L.M.

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulos 0 e 1.
Programação Avançada em MSX - capítulo 1.

7.K - ROTACIONANDO CARACTERES

A rotina apresentada a seguir permite a escrita com letras 'deitadas' na tela. Melhor do que tentar entender o que o programa faz, é rodá-lo e obervar o efeito.

O programa em BASIC serve para gerar uma rotina em Linguagem de Máquina. Ao ser chamada por uma instrução USR, o caractere cujo código é passado como parâmetro será "girado" de 90°.

Analise as linhas de 200 a 240 para entender melhor como a rotina deve ser usada.

```
10 REM                                    99
20 REM GIRA CARACTERES NA SCREEN 1        918
30 REM                                    A26
100 CLEAR 300,&HCFFF                      DAE
110 SCREEN 1                              F79
120 DEFINT A                              111D
130 DEFUSR0=&HD000                        159E
140 FOR F=&HD000 TO &HD033                1C6A
150   READ A$:A=VAL("&H"+A$)              2567
160   POKE F,A                            2929
170 NEXT F                                2BF0
180 INPUT "MENSAGEM";A$                   3589
190 PRINT                                 3663
200 FOR F=1 TO LEN(A$)                    3EE0
210   A=ASC(MID$(A$,F,1))*8               4A04
220   PRINT CHR$(A/8)                     515E
230   X=USR0(A)                           583E
240 NEXT F                                5B00
250 GOTO 250                              5F26
260 REM                                   6126
270 DATA 23,23,5E,23,56,62,6B,E5          771E
280 DATA 01,08,00,11,2C,D0,D5,CD          8480
290 DATA 59,00,D1,E1,06,08,C5,06          8964
300 DATA 08,D5,EB,CB,26,CB,1F,23          8F80
310 DATA 10,F9,EB,CD,4D,00,23,D1          9909
320 DATA C1,10,EB,C9,00,00,00,00          A110
330 DATA 00,00,00,00                      A9EC

                          TOTAL   =       A9EC
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - páginas 140 a 143.

7.L - ROTINA DE ENTRADA COM INKEY$

A rotina apresentada a seguir permite a introdução de dados em programas através da instrução INKEY$.

As linhas de 100 a 170 simulam um programa qualquer. A única linha realmente necessária é a linha 110, onde uma função-string para posicionamento do cursor é definida. A rotina de entrada efetivamente começa na linha 1000.

Ao ser chamada, a sub-rotina da linha 1000 deve receber a LINHA (na variável LI) e a COLUNA (na variável CO) em que os dados introduzidos deverão aparecer na tela. Além disso deve-se também fornecer quantos caracteres poderão ser introduzidos, isto é, o TAMANHO DA LINHA (na variável TL) a ser introduzida.

Para entrar dados com essa rotina, dispõe-se das seguintes funções:

RETURN - termina a inserção de dados;
BS ou - Volta uma posição apagando;
CLS - Apaga a linha já introduzida.

Ao retornar da sub-rotina (quando se digita RETURN), o programa tráz na variável L$ a linha introduzida.

Essa rotina pode ser particularmente útil nos programas de gerenciamento de dados, em que ao serem introduzidas, as informações de cada campo devem ser apresentadas de forma estética na tela.

```
100 SCREEN 0 : WIDTH 39
110 DEF FNPC$(LI,CO)=CHR$(27)+"Y"+
       CHR$(LI+32)+CHR$(CO+32)
120 INPUT "LINHA:";LI
130 INPUT "COLUNA:";CO
140 INPUT "Nº CARACTERES:";TL
150 GOSUB 1000
160 PRINT : PRINT,,,,,,,,"L$=";L$
170 IF STRIG(0) THEN RUN ELSE 170
1000 '
1010 ' SUB-ROTINA PARA ENTRAR LINHA
1020 '
1030 L$ = ""
1040 PRINT FNPC$(LI,CO);"_";
1050 I$=INKEY$
1060 IF I$="" THEN 1050
1070 IF I$>=" " AND I$<="{" THEN 1230
1080 C=CO+LEN(L$):PRINT FNPC$(LI,C);
```

```
1090 IF LEN(L$)<TL THEN PRINT " "            9485
1100 I=ASC(I$)                                9B7A
1110 IF I <>  8 THEN 1170          ' BS       A689
1120   IF LEN(L$)=0 THEN 1140                 B1F0
1130     L$=LEFT$(L$,LEN(L$)-1)               C012
1140   C = CO : PRINT FNPC$(LI,C);            D21F
1150   PRINT L$;"_"                           D88A
1160   GOTO 1050                              E25B
1170 IF I <> 12 THEN 1220          'CLS       F679
1180   L$=""                                  FB34
1190   PRINT FNPC$(LI,CO);"_";                198
1200   PRINT STRING$(TL-1," ")                642
1210   GOTO 1050                              870
1220 IF I = 13 THEN RETURN         'RETURN    12FA
1230 IF LEN(L$)<TL THEN L$=L$+I$              1C04
1240 IF LEN(L$)=TL THEN L$=LEFT$(L$,TL-1      2EA2
)+I$
1250 C=CO-1+LEN(L$)                           3729
1260 PRINT FNPC$(LI,C);                       4074
1270 PRINT I$;                                4449
1280 IF LEN(L$)<TL THEN PRINT"_"              502B
1290 GOTO 1050                                5556

                              TOTAL    =      5556
```

7.M - ROTINA PARA MAIÚSCULAS

O programa apresentado abaixo gera uma pequena rotina em Linguagem de Máquina que passa todas as letras comuns minúsculas de uma string para maiúsculas.

Experimente rodá-lo e introduza algumas sequências de letras minúsculas. Note que a string é passada como parâmetro da função USR.

```
100 CLEAR 300,&HBFFF                          33F
110 FOR I = &HC000 TO &HC021                  97B
120     READ X$                               B6F
130     POKE I,VAL("&h"+X$)                  1128
140 NEXT I                                   12FA
150 DATA 3A,63,F6,FE,03,C0,2A,F8             1CA3
160 DATA F7,7E,B7,C8,47,23,5E,23             25AA
170 DATA 66,6B,7E,FE,61,38,07,FE             36A5
180 DATA 7B,30,03,D6,20,77,23,10             46A8
190 DATA F1,C9                               48C3
200 DEFUSR0=&HC000                           5075
210 INPUT X$                                 5443
220 PRINT USR0(X$)                           5A96
230 GOTO 210                                 5E91

                              TOTAL   =      5E91
```

7.N - REDUZINDO A TELA A UM CARACTERE

Você sabe que podemos usar o comando WIDTH para as telas de texto de modo a fazê-las ficar com apenas uma coluna. Experimente comandar:

```
WIDTH 1
```

Agora a tela tem apenas uma coluna. Para voltar ao normal, basta usar novamente o comando WIDTH.

Um fato menos conhecido é a possibilidade de fazer com que a tela tenha apenas 1 linha. Para isso, entretanto, não existe um comando dedicado do BASIC e é necessário usar o comando POKE para alterar o valor de uma variável do sistema: a CRTCNT, em &HF3B1.

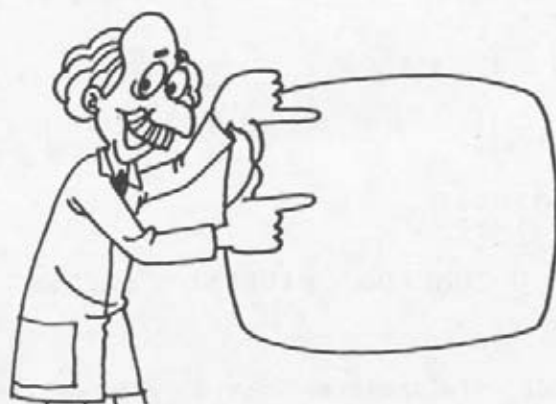
Experimente comandar:

```
POKE &HF3B1,1
```

Com isso a tela deverá ficar com apenas uma linha.

Para reduzir a tela a apenas um caractere, basta usar o par de comandos:

```
WIDTH1:POKE&HF3B1,1
```

Isso, entretanto, tornará o teclado inoperante e você perderá o controle sobre o micro. Portanto, cuidado ao usar esta dica !

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - pagina 47.

7.0 - GRANDES EXPOENTES

Recentemente, astrônomos americanos detectaram o que parecem ser galáxias em formação a 17 bilhões de anos-luz da Terra! Estes objetos estariam quase no limite do Universo observável, cujo raio, hoje, é estimado em 20 bilhões de anos-luz.

Aproveitando que você tem um computador na frente, você resolve calcular o raio do Universo em metros.

O cálculo é simples (pelo menos no computador!); você sabe que a luz se propaga com uma velocidade de 300.000 quilômetros por segundo. Basta portanto multiplicar esta velocidade de 300.000.000 m/s pelo número de segundos em 20 bilhões de anos (lembre-se que o ano tem 365 dias de 24 horas de 60 minutos de 60 segundos).

Digite, então, o seguinte programinha:

```
10 R=300000000#*20000000000#*365*24*60*6
0
20 PRINT"RAIO DO UNIVERSO=";R;"METROS"
```

Rodando-o, você deve obter a considerável quantia de 1,89216 vezes 10 elevado à potência 26!

Entusiasmado com a rapidez de cálculo do seu MSX, você resolve calcular o volume do Universo (supondo ingenuamente que ele seja esférico).

Lembrando que o volume de uma esfera é dado pela expressão:

$$V = 4/3.\pi.R^3$$

Você, então, completa seu programinha com as seguintes linhas:

```
30 PI=4*ATN(1)
40 X=(4/3)*PI
50 V=X*R^3
60 PRINT"VOLUME DO UNIVERSO=";V;"m3"
```

Ao rodar seu programa assim incrementado, você tem a decepção de obter um "overflow in 50", pois você "estourou" a capacidade de cálculo do MSX.

Nesta dica, vamos apresentar como contornar o problema de expoentes grandes demais. Basta lembrar

que:

log (AxB) = log A + log B

log A^n = n x log A

Substitua as linhas 50 e 60 por:

```
100 LV=LOG(X)+3*LOG(R)                     2366
110 LD=LV/LOG(10)                          28EB
120 E=INT(LD)                              2E0B
130 M=10^(LD-E)                            3599
140 A$=STR$(M)+"E+"+STR$(E)                4086
150 PRINT"VOL. DO UNIVERSO=";A$;"M.C."     599E
```

Na linha 100 você calcula o logarítmo neperiano do volume (o MSX só trabalha com logarítmos naturais). Na linha 110 você transforma o log natural em log decimal (é só dividir por LOG(10): esta é mais uma dica importante).

Para entender as linhas 120 e 130, lembre-se que se, por exemplo,

$$\log x = 79,4529$$

então

$$x = 10^{0,4529} * 10^{79}$$

Portanto, toda vez que você se defrontar com números demasiadamente grandes para seu MSX, basta calcular seu logaritmo decimal, pegar a parte inteira com expoente de 10 e descobrir o multiplicando na frente dele, operando como aprendemos.

NOTAS SOBRE A BIBLIOGRAFIA RECOMENDADA

Como você deve ter percebido, as dicas deste livro foram apresentadas de forma bem prática e resumida, de modo a poderem ser usadas imediatamente. Se você quiser obter maiores detalhes sobre o funcionamento de cada uma delas é recomendável estudar detalhadamente os livros citados na "BIBLIOGRAFIA RECOMENDADA", ao final de cada dica. A seguir apresentamos um resumo do conteúdo de cada um desses livros para melhor orientá-lo.

LINGUAGEM BASIC MSX
Uma "enciclopédia" do BASIC MSX, com a sintaxe, função e exemplo de cada palavra do BASIC MSX.
CURSO DE BASIC MSX v.1
Uma introdução clara e didática ao BASIC residente do MSX, apresentada em 8 aulas com exercícios (e suas respostas!).
COLEÇÃO DE PROGRAMAS PARA MSX v.1 e v.2
Programas didáticos, aplicativos e utilitários explicados passo a passo para que o leitor aprenda a fazer seus próprios programas.
APROFUNDANDO-SE NO MSX
O "best seller" da literatura técnica sobre MSX, com a descrição detalhada da arquitetura da máquina e de cada uma de suas partes.
PROGRAMAÇÃO AVANÇADA EM MSX
Exemplos e rotinas utilitárias em ASSEMBLY ensinando ao leitor como se obtém o máximo das máquinas MSX.
LINGUAGEM DE MÁQUINA MSX
Uma introdução completa e didática aos poderosos recursos da Linguagem de Máquina Z80 aplicada aos micros MSX. Contém as instruções secretas do Z80.
USANDO O DISK DRIVE NO MSX
O MSXDOS, O CP/M para MSX e o DISK BASIC comentados exaustivamente.
SISTEMA DE DISCO PARA MSX
O SOLXDOS e o BASIC de DISCO comentados passo a passo de forma clara e didática.
DRIVES LEOPARD DE 3 1/2"
O primeiro livro sobre drives de 3 1/2" editado no Brasil. Contém todos os recursos do MSXDOS e do DISK BASIC MSX aplicados aos drives de 3 1/2".

Para receber gratuitamente o boletim informativo da ALEPH, contendo dicas de programação, artigos técnicos e informações sobre os últimos lançamentos para seu micro, envie seu nome e endereço completos (incluindo o CEP) para:

EDITORA ALEPH
Caixa Postal: 20.707
01498 São Paulo SP

Se você quiser adquirir todos os programas com mais de 512 bytes listados neste livro já gravados em DISCO (apenas em disco!!!), entre em contato conosco.

Para comprar nossos livros pelo correio, informe-se escrevendo ou telefonando para nós.

Nosso telefone é:

(011) 843-3202

Impressão e acabamento
(com filmes fornecidos):
EDITORA SANTUÁRIO
Fone (0125) 36-2140
APARECIDA - SP

COLEÇÃO MSX

1ØØ DICAS PARA MSX

Ao longo de dois anos trabalhando com micros MSX pudemos atender a milhares (sem exagero!) de dúvidas sobre essas máquinas.

Esquentando a orelha ao telefone, atolando em montanhas de cartas, ou atendendo pessoalmente nossos leitores, surgiram as mais de cem dicas publicadas neste livro, como uma espécie de resposta coletiva.

Apesar de apelarmos frequentemente para o uso da Linguagem de Máquina, todos os programas listados neste livro estão em BASIC, prontos para serem usados e com checagem automática para detecção de erros de digitação.

Esperamos com isso abrir novos horizontes aos nossos leitores, tanto aos mais experientes quanto aos principiantes.